SpaceStone

Smart Contract Audit Report

SYSFIXED

08 May 2021

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
SpaceStone	SpaceStone	Binance Smart Chain

Addresses

Contract address	0xd778eb7f995955fcf0ee746bc0f20c1729252704
Contract deployer address	0xE05f2F02896114ad3c9032212fa74Eaad8eac021

Project Website

https://spacestone.org/

Codebase

https://bscscan.com/address/0xd778eb7f995955fcf0ee746bc0f20c1729252704#code

SUMMARY

Space Stone token has a unique idea and will be the most valuable asset for use in science, technology and space tech economy in Mars and the Moon

Contract Summary

Documentation Quality

SpaceStone 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 SpaceStone 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 722.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 110, 142, 165, 166, 201, 237, 464, 705, 705, 705, 705, 706, 706, 725, 725, 725, 725, 726, 726, 726, 726, 857, 859, 896, 942, 961, 967 and 859.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 11.
- 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 858, 859, 859, 943, 943, 944, 945, 1070 and 1071.

CONCLUSION

We have audited the SpaceStone project released on May 2021 to discover issues and identify potential security vulnerabilities in SpaceStone Project. This process is used to find technical issues and security loopholes which might be found in the smart contract.

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

The issues found in the SpaceStone smart contract code do not pose a considerable risk. The writing of the contract is close to the standard of writing contracts in general. The low-risk issues found are some arithmetic operation issues, a floating pragma is set, 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.	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	Friday May 07 2021 12:11:17 GMT+0000 (Coordinated Universal Time)
Finished	Saturday May 08 2021 13:18:15 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	SpaceStone.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-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 110

low SEVERITY

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

Source File

- SpaceStone.sol

Locations

```
109   function add(uint256 a, uint256 b) internal pure returns (uint256) {
110   uint256 c = a + b;
111   require(c >= a, "SafeMath: addition overflow");
112
113   return c;
114
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 142

low SEVERITY

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

Source File

- SpaceStone.sol

Locations

```
141   require(b <= a, errorMessage);
142   uint256 c = a - b;
143
144   return c;
145   }
146
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 165

low SEVERITY

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

Source File

- SpaceStone.sol

Locations

```
164
165   uint256 c = a * b;
166   require(c / a == b, "SafeMath: multiplication overflow");
167
168   return c;
169
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 166

low SEVERITY

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

Source File

- SpaceStone.sol

Locations

```
165   uint256 c = a * b;
166   require(c / a == b, "SafeMath: multiplication overflow");
167
168   return c;
169   }
170
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 201

low SEVERITY

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

Source File

- SpaceStone.sol

Locations

```
200   require(b > 0, errorMessage);
201   uint256 c = a / b;
202   // assert(a == b * c + a % b); // There is no case in which this doesn't hold
203
204   return c;
205
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 237

low SEVERITY

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

Source File

- SpaceStone.sol

Locations

```
236   require(b != 0, errorMessage);
237   return a % b;
238   }
239   }
240
241
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 464

low SEVERITY

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

Source File

- SpaceStone.sol

Locations

```
463   _owner = address(0);
464   _lockTime = now + time;
465   emit OwnershipTransferred(_owner, address(0));
466   }
467
468
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 705

low SEVERITY

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

Source File

- SpaceStone.sol

Locations

```
704   uint256 private constant MAX = ~uint256(0);
705   uint256 private _tTotal = 1000000000 * 10**6 * 10**9;
706   uint256 private _rTotal = (MAX - (MAX % _tTotal));
707   uint256 private _tFeeTotal;
708
709
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 705

low SEVERITY

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

Source File

- SpaceStone.sol

Locations

```
704   uint256 private constant MAX = ~uint256(0);
705   uint256 private _tTotal = 1000000000 * 10**6 * 10**9;
706   uint256 private _rTotal = (MAX - (MAX % _tTotal));
707   uint256 private _tFeeTotal;
708
709
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 705

low SEVERITY

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

Source File

- SpaceStone.sol

Locations

```
704   uint256 private constant MAX = ~uint256(0);
705   uint256 private _tTotal = 1000000000 * 10**6 * 10**9;
706   uint256 private _rTotal = (MAX - (MAX % _tTotal));
707   uint256 private _tFeeTotal;
708
709
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 705

low SEVERITY

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

Source File

- SpaceStone.sol

Locations

```
704   uint256 private constant MAX = ~uint256(0);
705   uint256 private _tTotal = 1000000000 * 10**6 * 10**9;
706   uint256 private _rTotal = (MAX - (MAX % _tTotal));
707   uint256 private _tFeeTotal;
708
709
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 706

low SEVERITY

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

Source File

- SpaceStone.sol

Locations

```
705   uint256 private _tTotal = 1000000000 * 10**6 * 10**9;
706   uint256 private _rTotal = (MAX - (MAX % _tTotal));
707   uint256 private _tFeeTotal;
708
709   string private _name = "SpaceStone";
710
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 706

low SEVERITY

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

Source File

- SpaceStone.sol

Locations

```
705   uint256 private _tTotal = 1000000000 * 10**6 * 10**9;
706   uint256 private _rTotal = (MAX - (MAX % _tTotal));
707   uint256 private _tFeeTotal;
708
709   string private _name = "SpaceStone";
710
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 725

low SEVERITY

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

Source File

- SpaceStone.sol

Locations

```
724
725   uint256 public _maxTxAmount = 5000000 * 10**6 * 10**9;
726   uint256 private numTokensSellToAddToLiquidity = 500000 * 10**6 * 10**9;
727
728   event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
729
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 725

low SEVERITY

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

Source File

- SpaceStone.sol

Locations

```
724
725   uint256 public _maxTxAmount = 5000000 * 10**6 * 10**9;
726   uint256 private numTokensSellToAddToLiquidity = 500000 * 10**6 * 10**9;
727
728   event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
729
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 725

low SEVERITY

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

Source File

- SpaceStone.sol

Locations

```
724
725   uint256 public _maxTxAmount = 5000000 * 10**6 * 10**9;
726   uint256 private numTokensSellToAddToLiquidity = 500000 * 10**6 * 10**9;
727
728   event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
729
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 725

low SEVERITY

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

Source File

- SpaceStone.sol

Locations

```
724
725   uint256 public _maxTxAmount = 5000000 * 10**6 * 10**9;
726   uint256 private numTokensSellToAddToLiquidity = 500000 * 10**6 * 10**9;
727
728   event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
729
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 726

low SEVERITY

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

Source File

- SpaceStone.sol

Locations

```
725   uint256 public _maxTxAmount = 5000000 * 10**6 * 10**9;
726   uint256 private numTokensSellToAddToLiquidity = 500000 * 10**6 * 10**9;
727
728   event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
729   event SwapAndLiquifyEnabledUpdated(bool enabled);
730
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 726

low SEVERITY

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

Source File

- SpaceStone.sol

Locations

```
725   uint256 public _maxTxAmount = 5000000 * 10**6 * 10**9;
726   uint256 private numTokensSellToAddToLiquidity = 500000 * 10**6 * 10**9;
727
728   event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
729   event SwapAndLiquifyEnabledUpdated(bool enabled);
730
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 726

low SEVERITY

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

Source File

- SpaceStone.sol

Locations

```
725   uint256 public _maxTxAmount = 5000000 * 10**6 * 10**9;
726   uint256 private numTokensSellToAddToLiquidity = 500000 * 10**6 * 10**9;
727
728   event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
729   event SwapAndLiquifyEnabledUpdated(bool enabled);
730
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 726

low SEVERITY

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

Source File

- SpaceStone.sol

Locations

```
725   uint256 public _maxTxAmount = 5000000 * 10**6 * 10**9;
726   uint256 private numTokensSellToAddToLiquidity = 500000 * 10**6 * 10**9;
727
728   event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
729   event SwapAndLiquifyEnabledUpdated(bool enabled);
730
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 857

low SEVERITY

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

Source File

- SpaceStone.sol

Locations

```
856   require(_isExcluded[account], "Account is already excluded");
857   for (uint256 i = 0; i < _excluded.length; i++) {
858   if (_excluded[i] == account) {
859   _excluded[i] = _excluded[_excluded.length - 1];
860   _tOwned[account] = 0;
861
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 859

low SEVERITY

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

Source File

- SpaceStone.sol

Locations

```
858   if (_excluded[i] == account) {
859   _excluded[i] = _excluded[_excluded.length - 1];
860   _tOwned[account] = 0;
861   _isExcluded[account] = false;
862   _excluded.pop();
863
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 896

low SEVERITY

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

Source File

- SpaceStone.sol

Locations

```
895   _maxTxAmount = _tTotal.mul(maxTxPercent).div(
896   10**2
897   );
898   }
899
900
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 942

low SEVERITY

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

Source File

- SpaceStone.sol

Locations

```
941   uint256 tSupply = _tTotal;
942   for (uint256 i = 0; i < _excluded.length; i++) {
943   if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return
(_rTotal, _tTotal);
944   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
945   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
946
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 961

low SEVERITY

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

Source File

- SpaceStone.sol

Locations

```
960   return _amount.mul(_taxFee).div(
961   10**2
962   );
963   }
964
965
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 967

low SEVERITY

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

Source File

- SpaceStone.sol

Locations

```
966   return _amount.mul(_liquidityFee).div(
967   10**2
968   );
969   }
970
971
```

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

LINE 859

low SEVERITY

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

Source File

- SpaceStone.sol

Locations

```
858   if (_excluded[i] == account) {
859   _excluded[i] = _excluded[_excluded.length - 1];
860   _tOwned[account] = 0;
861   _isExcluded[account] = false;
862   _excluded.pop();
863
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 11

low SEVERITY

The current pragma Solidity directive is ""^0.6.12"". 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

- SpaceStone.sol

Locations

```
10
11   pragma solidity ^0.6.12;
12   // SPDX-License-Identifier: Unlicensed
13   interface IERC20 {
14
15
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 722

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

- SpaceStone.sol

Locations

```
721
722   bool inSwapAndLiquify;
723   bool public swapAndLiquifyEnabled = true;
724
725   uint256 public _maxTxAmount = 5000000 * 10**6 * 10**9;
726
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 858

low SEVERITY

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

Source File

- SpaceStone.sol

Locations

```
857   for (uint256 i = 0; i < _excluded.length; i++) {
858   if (_excluded[i] == account) {
859   _excluded[i] = _excluded[_excluded.length - 1];
860   _tOwned[account] = 0;
861   _isExcluded[account] = false;
862
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 859

low SEVERITY

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

Source File

- SpaceStone.sol

Locations

```
858   if (_excluded[i] == account) {
859   _excluded[i] = _excluded[_excluded.length - 1];
860   _tOwned[account] = 0;
861   _isExcluded[account] = false;
862   _excluded.pop();
863
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 859

low SEVERITY

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

Source File

- SpaceStone.sol

Locations

```
858   if (_excluded[i] == account) {
859   _excluded[i] = _excluded[_excluded.length - 1];
860   _tOwned[account] = 0;
861   _isExcluded[account] = false;
862   _excluded.pop();
863
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 943

low SEVERITY

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

Source File

- SpaceStone.sol

Locations

```
942   for (uint256 i = 0; i < _excluded.length; i++) {
943   if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return
(_rTotal, _tTotal);
944   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
945   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
946   }
947
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 943

low SEVERITY

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

Source File

- SpaceStone.sol

Locations

```
942   for (uint256 i = 0; i < _excluded.length; i++) {
943   if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return
(_rTotal, _tTotal);
944   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
945   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
946   }
947
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 944

low SEVERITY

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

Source File

- SpaceStone.sol

Locations

```
943   if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return
(_rTotal, _tTotal);
944   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
945   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
946   }
947   if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
948
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 945

low SEVERITY

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

Source File

- SpaceStone.sol

Locations

```
944   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
945   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
946   }
947   if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
948   return (rSupply, tSupply);
949
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1070

low SEVERITY

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

Source File

- SpaceStone.sol

Locations

```
1069   address[] memory path = new address[](2);
1070   path[0] = address(this);
1071   path[1] = uniswapV2Router.WETH();
1072
1073   _approve(address(this), address(uniswapV2Router), tokenAmount);
1074
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1071

low SEVERITY

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

Source File

- SpaceStone.sol

Locations

```
1070   path[0] = address(this);
1071   path[1] = uniswapV2Router.WETH();
1072
1073   _approve(address(this), address(uniswapV2Router), tokenAmount);
1074
1075
```

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.