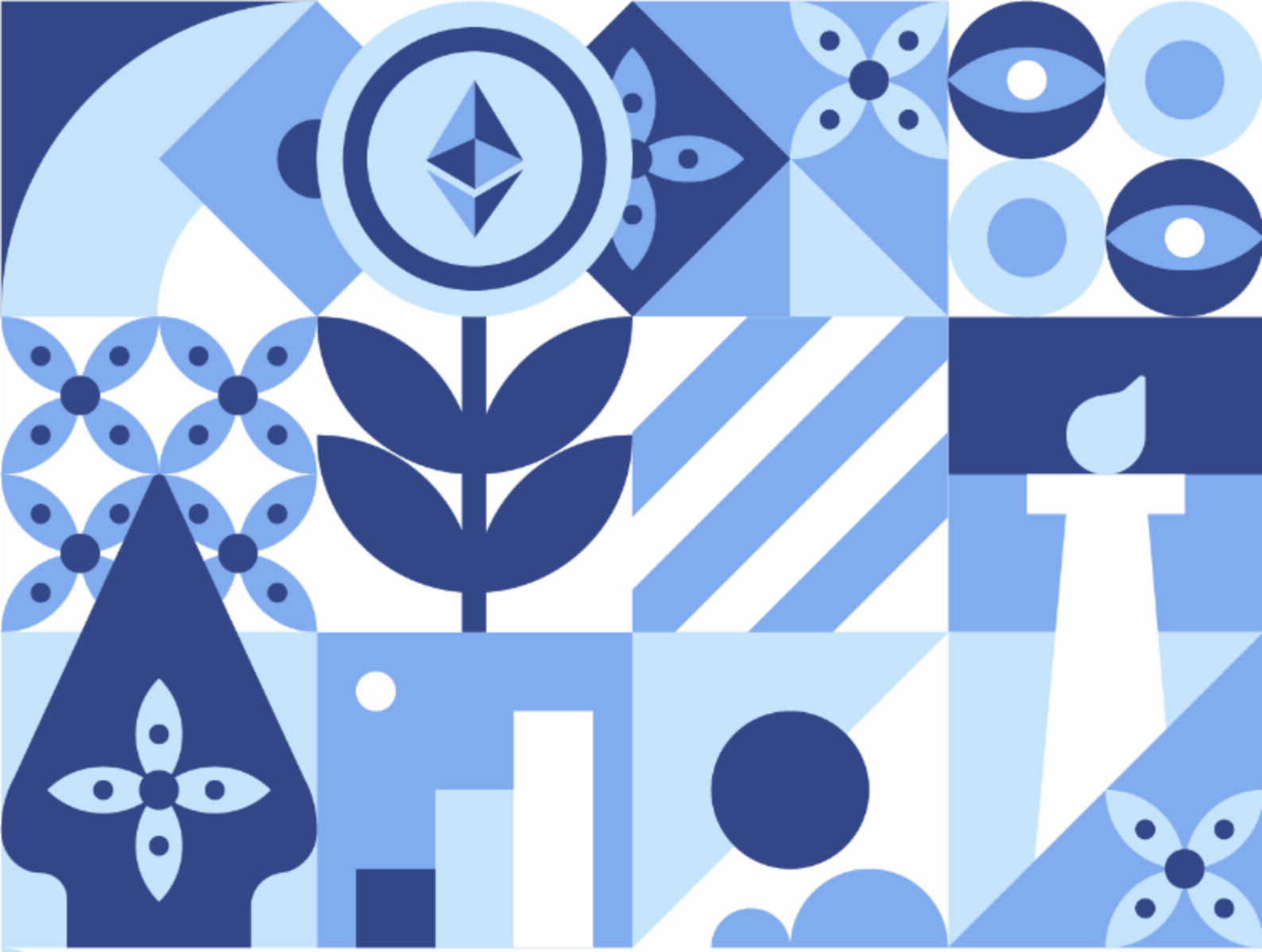

NIL Coin

Smart Contract Audit Report

SYSFIXED

26 Nov 2021

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
NIL Coin	NIL	Ethereum

Addresses

Contract address	0xD31B00deA80cF282aCE1791D204d76a85Fb82556
Contract deployer address	0x51049066d8ce32D647c6e5E5a92E037040a7f70D

Project Website

https://nilcoin.com/

Codebase

https://etherscan.io/address/0xD31B00deA80cF282aCE1791D204d76a85Fb82556#code

SUMMARY

NIL is the Official Crypto that rewards College Athletes for their name, image & likeness when promoting products and services on social media.

Contract Summary

Documentation Quality

NIL Coin 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 NIL Coin with the discovery of several low issues.

Test Coverage

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

Audit Findings Summary

- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 134, 170, 193, 194, 233, 273, 546, 929, 929, 929, 929, 930, 930, 954, 954, 954, 954, 955, 955, 955, 955, 1097, 1097, 1114, 1114, 1115, 1171, 1173, 1241, 1241, 1252, 1252, 1371, 1392, 1400, 1400, 1488 and 1173.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 24.
- 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 1114, 1114, 1172, 1173, 1173, 1373, 1374, 1376, 1377, 1524 and 1525.

CONCLUSION

We have audited the NIL Coin project released on November 2021 to discover issues and identify potential security vulnerabilities in NIL Coin 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 NIL Coin 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 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.	PASS
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	Thursday Nov 25 2021 17:13:14 GMT+0000 (Coordinated Universal Time)
Finished	Friday Nov 26 2021 23:27:39 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	NIL.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	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged
SWC-103	A FLOATING PRAGMA IS 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-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 134

low SEVERITY

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

Source File

- NIL.sol

Locations

```
133   function add(uint256 a, uint256 b) internal pure returns (uint256) {
134   uint256 c = a + b;
135   require(c >= a, "SafeMath: addition overflow");
136
137   return c;
138
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 170

low SEVERITY

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

Source File

- NIL.sol

Locations

```
169   require(b <= a, errorMessage);
170   uint256 c = a - b;
171
172   return c;
173   }
174
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 193

low SEVERITY

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

Source File

- NIL.sol

Locations

```
192
193   uint256 c = a * b;
194   require(c / a == b, "SafeMath: multiplication overflow");
195
196   return c;
197
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 194

low SEVERITY

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

Source File

- NIL.sol

Locations

```
193   uint256 c = a * b;
194   require(c / a == b, "SafeMath: multiplication overflow");
195
196   return c;
197   }
198
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 233

low SEVERITY

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

Source File

- NIL.sol

Locations

```
232   require(b > 0, errorMessage);
233   uint256 c = a / b;
234   // assert(a == b * c + a % b); // There is no case in which this doesn't hold
235
236   return c;
237
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 273

low SEVERITY

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

Source File

- NIL.sol

Locations

```
272   require(b != 0, errorMessage);
273   return a % b;
274   }
275   }
276
277
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 546

low SEVERITY

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

Source File

- NIL.sol

Locations

```
545   _owner = address(0);
546   _lockTime = block.timestamp + time;
547   emit OwnershipTransferred(_owner, address(0));
548   }
549
550
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 929

low SEVERITY

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

Source File

- NIL.sol

Locations

```
928   uint256 private constant MAX = ~uint256(0);
929   uint256 private _tTotal = 1000000000 * 10**3 * 10**8;
930   uint256 private _rTotal = (MAX - (MAX % _tTotal));
931   uint256 private _tFeeTotal;
932
933
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 929

low SEVERITY

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

Source File

- NIL.sol

Locations

```
928   uint256 private constant MAX = ~uint256(0);
929   uint256 private _tTotal = 1000000000 * 10**3 * 10**8;
930   uint256 private _rTotal = (MAX - (MAX % _tTotal));
931   uint256 private _tFeeTotal;
932
933
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 929

low SEVERITY

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

Source File

- NIL.sol

Locations

```
928   uint256 private constant MAX = ~uint256(0);
929   uint256 private _tTotal = 1000000000 * 10**3 * 10**8;
930   uint256 private _rTotal = (MAX - (MAX % _tTotal));
931   uint256 private _tFeeTotal;
932
933
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 929

low SEVERITY

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

Source File

- NIL.sol

Locations

```
928   uint256 private constant MAX = ~uint256(0);
929   uint256 private _tTotal = 1000000000 * 10**3 * 10**8;
930   uint256 private _rTotal = (MAX - (MAX % _tTotal));
931   uint256 private _tFeeTotal;
932
933
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 930

low SEVERITY

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

Source File

- NIL.sol

Locations

```
929   uint256 private _tTotal = 1000000000 * 10**3 * 10**8;
930   uint256 private _rTotal = (MAX - (MAX % _tTotal));
931   uint256 private _tFeeTotal;
932
933   address public burnAddress = 0x000000000000000000000000000000000000dEaD;
934
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 930

low SEVERITY

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

Source File

- NIL.sol

Locations

```
929   uint256 private _tTotal = 1000000000 * 10**3 * 10**8;
930   uint256 private _rTotal = (MAX - (MAX % _tTotal));
931   uint256 private _tFeeTotal;
932
933   address public burnAddress = 0x000000000000000000000000000000000000dEaD;
934
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 954

low SEVERITY

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

Source File

- NIL.sol

Locations

```
953
954   uint256 public _maxTxAmount = 2000000 * 10**3 * 10**8;
955   uint256 public numTokensSellToAddToLiquidity = 1000000 * 10**3 * 10**8;
956
957   event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
958
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 954

low SEVERITY

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

Source File

- NIL.sol

Locations

```
953
954   uint256 public _maxTxAmount = 2000000 * 10**3 * 10**8;
955   uint256 public numTokensSellToAddToLiquidity = 1000000 * 10**3 * 10**8;
956
957   event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
958
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 954

low SEVERITY

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

Source File

- NIL.sol

Locations

```
953
954   uint256 public _maxTxAmount = 2000000 * 10**3 * 10**8;
955   uint256 public numTokensSellToAddToLiquidity = 1000000 * 10**3 * 10**8;
956
957   event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
958
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 954

low SEVERITY

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

Source File

- NIL.sol

Locations

```
953
954   uint256 public _maxTxAmount = 2000000 * 10**3 * 10**8;
955   uint256 public numTokensSellToAddToLiquidity = 1000000 * 10**3 * 10**8;
956
957   event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
958
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 955

low SEVERITY

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

Source File

- NIL.sol

Locations

```
954   uint256 public _maxTxAmount = 2000000 * 10**3 * 10**8;
955   uint256 public numTokensSellToAddToLiquidity = 1000000 * 10**3 * 10**8;
956
957   event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
958   event SwapAndLiquifyEnabledUpdated(bool enabled);
959
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 955

low SEVERITY

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

Source File

- NIL.sol

Locations

```
954   uint256 public _maxTxAmount = 2000000 * 10**3 * 10**8;
955   uint256 public numTokensSellToAddToLiquidity = 1000000 * 10**3 * 10**8;
956
957   event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
958   event SwapAndLiquifyEnabledUpdated(bool enabled);
959
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 955

low SEVERITY

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

Source File

- NIL.sol

Locations

```
954   uint256 public _maxTxAmount = 2000000 * 10**3 * 10**8;
955   uint256 public numTokensSellToAddToLiquidity = 1000000 * 10**3 * 10**8;
956
957   event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
958   event SwapAndLiquifyEnabledUpdated(bool enabled);
959
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 955

low SEVERITY

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

Source File

- NIL.sol

Locations

```
954   uint256 public _maxTxAmount = 2000000 * 10**3 * 10**8;
955   uint256 public numTokensSellToAddToLiquidity = 1000000 * 10**3 * 10**8;
956
957   event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
958   event SwapAndLiquifyEnabledUpdated(bool enabled);
959
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1097

low SEVERITY

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

Source File

- NIL.sol

Locations

```
1096   removeAllFee();
1097   _transfer(_msgSender(), recipient, amount * 10**8);
1098   restoreAllFee();
1099   }
1100
1101
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1097

low SEVERITY

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

Source File

- NIL.sol

Locations

```
1096   removeAllFee();
1097   _transfer(_msgSender(), recipient, amount * 10**8);
1098   restoreAllFee();
1099   }
1100
1101
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1114

low SEVERITY

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

Source File

- NIL.sol

Locations

```
1113   while (iterator < newholders.length) {
1114   airdropInternal(newholders[iterator], amounts[iterator] * 10**8);
1115   iterator += 1;
1116   }
1117   }
1118
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1114

low SEVERITY

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

Source File

- NIL.sol

Locations

```
1113   while (iterator < newholders.length) {
1114   airdropInternal(newholders[iterator], amounts[iterator] * 10**8);
1115   iterator += 1;
1116   }
1117   }
1118
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1115

low SEVERITY

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

Source File

- NIL.sol

Locations

```
1114   airdropInternal(newholders[iterator], amounts[iterator] * 10**8);
1115   iterator += 1;
1116   }
1117   }
1118
1119
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1171

low SEVERITY

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

Source File

- NIL.sol

Locations

```
1170   require(_isExcluded[account], "Account is already excluded");
1171   for (uint256 i = 0; i < _excluded.length; i++) {
1172   if (_excluded[i] == account) {
1173   _excluded[i] = _excluded[_excluded.length - 1];
1174   _tOwned[account] = 0;
1175
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1173

low SEVERITY

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

Source File

- NIL.sol

Locations

```
1172   if (_excluded[i] == account) {
1173   _excluded[i] = _excluded[_excluded.length - 1];
1174   _tOwned[account] = 0;
1175   _isExcluded[account] = false;
1176   _excluded.pop();
1177
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1241

low SEVERITY

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

Source File

- NIL.sol

Locations

```
1240   );
1241   _maxTxAmount = maxTxAmount * 10**8;
1242   }
1243
1244   function setSwapThresholdAmount(uint256 SwapThresholdAmount)
1245
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1241

low SEVERITY

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

Source File

- NIL.sol

Locations

```
1240   );
1241   _maxTxAmount = maxTxAmount * 10**8;
1242   }
1243
1244   function setSwapThresholdAmount(uint256 SwapThresholdAmount)
1245
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1252

low SEVERITY

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

Source File

- NIL.sol

Locations

```
1251   );
1252   numTokensSellToAddToLiquidity = SwapThresholdAmount * 10**8;
1253   }
1254
1255   function claimTokens(address walletAddress) public onlyOwner {
1256
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1252

low SEVERITY

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

Source File

- NIL.sol

Locations

```
1251   );
1252   numTokensSellToAddToLiquidity = SwapThresholdAmount * 10**8;
1253   }
1254
1255   function claimTokens(address walletAddress) public onlyOwner {
1256
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1371

low SEVERITY

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

Source File

- NIL.sol

Locations

```
1370   uint256 tSupply = _tTotal;
1371   for (uint256 i = 0; i < _excluded.length; i++) {
1372   if (
1373   _rOwned[_excluded[i]] > rSupply ||
1374   _tOwned[_excluded[i]] > tSupply
1375
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1392

low SEVERITY

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

Source File

- NIL.sol

Locations

```
1391   function calculateTaxFee(uint256 _amount) private view returns (uint256) {
1392   return _amount.mul(_taxFee).div(10**2);
1393   }
1394
1395   function calculateLiquidityFee(uint256 _amount)
1396
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1400

low SEVERITY

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

Source File

- NIL.sol

Locations

```
1399   {
1400   return _amount.mul(_liquidityFee + _burnFee).div(10**2);
1401   }
1402
1403   function removeAllFee() private {
1404
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1400

low SEVERITY

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

Source File

- NIL.sol

Locations

```
1399   {
1400   return _amount.mul(_liquidityFee + _burnFee).div(10**2);
1401   }
1402
1403   function removeAllFee() private {
1404
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1488

low SEVERITY

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

Source File

- NIL.sol

Locations

```
1487   // split the contract balance to burnAmt and liquifyAmt
1488   uint256 totalFee = _liquidityFee + _burnFee;
1489   uint256 burnAmt = contractTokenBalance.mul(_burnFee).div(totalFee);
1490   uint256 liquifyAmt = contractTokenBalance.sub(burnAmt);
1491
1492
```

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

LINE 1173

low SEVERITY

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

Source File

- NIL.sol

Locations

```
1172   if (_excluded[i] == account) {
1173   _excluded[i] = _excluded[_excluded.length - 1];
1174   _tOwned[account] = 0;
1175   _isExcluded[account] = false;
1176   _excluded.pop();
1177
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 24

low SEVERITY

The current pragma Solidity directive is ""^0.8.9"". 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

- NIL.sol

Locations

```
23
24   pragma solidity ^0.8.9;
25
26   // SPDX-License-Identifier: Unlicensed
27   interface IERC20 {
28
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1114

low SEVERITY

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

Source File

- NIL.sol

Locations

```
1113   while (iterator < newholders.length) {
1114   airdropInternal(newholders[iterator], amounts[iterator] * 10**8);
1115   iterator += 1;
1116   }
1117   }
1118
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1114

low SEVERITY

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

Source File

- NIL.sol

Locations

```
1113   while (iterator < newholders.length) {
1114   airdropInternal(newholders[iterator], amounts[iterator] * 10**8);
1115   iterator += 1;
1116   }
1117   }
1118
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1172

low SEVERITY

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

Source File

- NIL.sol

Locations

```
1171   for (uint256 i = 0; i < _excluded.length; i++) {
1172   if (_excluded[i] == account) {
1173   _excluded[i] = _excluded[_excluded.length - 1];
1174   _tOwned[account] = 0;
1175   _isExcluded[account] = false;
1176
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1173

low SEVERITY

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

Source File

- NIL.sol

Locations

```
1172   if (_excluded[i] == account) {
1173   _excluded[i] = _excluded[_excluded.length - 1];
1174   _tOwned[account] = 0;
1175   _isExcluded[account] = false;
1176   _excluded.pop();
1177
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1173

low SEVERITY

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

Source File

- NIL.sol

Locations

```
1172   if (_excluded[i] == account) {
1173   _excluded[i] = _excluded[_excluded.length - 1];
1174   _tOwned[account] = 0;
1175   _isExcluded[account] = false;
1176   _excluded.pop();
1177
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1373

low SEVERITY

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

Source File

- NIL.sol

Locations

```
1372   if (
1373   _rOwned[_excluded[i]] > rSupply ||
1374   _tOwned[_excluded[i]] > tSupply
1375   ) return (_rTotal, _tTotal);
1376   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1377
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1374

low SEVERITY

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

Source File

- NIL.sol

Locations

```
1373   _rOwned[_excluded[i]] > rSupply ||
1374   _tOwned[_excluded[i]] > tSupply
1375   ) return (_rTotal, _tTotal);
1376   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1377   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
1378
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1376

low SEVERITY

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

Source File

- NIL.sol

Locations

```
1375   ) return (_rTotal, _tTotal);
1376   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1377   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
1378   }
1379   if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
1380
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1377

low SEVERITY

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

Source File

- NIL.sol

Locations

```
1376   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1377   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
1378   }
1379   if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
1380   return (rSupply, tSupply);
1381
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1524

low SEVERITY

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

Source File

- NIL.sol

Locations

```
1523   address[] memory path = new address[](2);
1524   path[0] = address(this);
1525   path[1] = uniswapV2Router.WETH();
1526
1527   _approve(address(this), address(uniswapV2Router), tokenAmount);
1528
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1525

low SEVERITY

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

Source File

- NIL.sol

Locations

```
1524   path[0] = address(this);
1525   path[1] = uniswapV2Router.WETH();
1526
1527   _approve(address(this), address(uniswapV2Router), tokenAmount);
1528
1529
```

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.