



Tora Inu

# Smart Contract Audit Report

# TABLE OF CONTENTS

## [Audited Details](#)

- Audited Project
- Blockchain
- Addresses
- Project Website
- Codebase

## [Summary](#)

- Contract Summary
- Audit Findings Summary
- Vulnerabilities Summary

## [Conclusion](#)

## [Audit Results](#)

## [Smart Contract Analysis](#)

- Detected Vulnerabilities

## [Disclaimer](#)

## [About Us](#)

# AUDITED DETAILS

## Audited Project

Project name	Token ticker	Blockchain
Tora Inu	TORA	Ethereum

## Addresses

Contract address	0x85E43525D41A33bCCB4bCF1Cb3f2ab0B74D59343
Contract deployer address	0x92A5E5FE9DEEe88A538c4f0e53dC129db900ACE2

## Project Website

<https://www.torainu.io/>

## Codebase

<https://etherscan.io/address/0x85E43525D41A33bCCB4bCF1Cb3f2ab0B74D59343#code>

# SUMMARY

Welcome to the next era of Inu Deflationary rewards meme token, P2E NFT game, and metaverse.

## Contract Summary

### Documentation Quality

Tora Inu provides a very good documentation with standard of solidity base code.

- The technical description is provided clearly and structured and also don't have any high risk issue.

### Code Quality

The Overall quality of the basecode is standard.

- Standard solidity basecode and rules are already followed by Tora Inu 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 561.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 121, 153, 176, 177, 212, 248, 475, 535, 535, 536, 536, 564, 564, 565, 565, 698, 700, 753, 784, 815, 836, 842, 848, 854, 909, 964 and 700.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 22.
- 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 699, 700, 700, 816, 816, 817, 818, 1002 and 1003.

## CONCLUSION

We have audited the Tora Inu project released on October 2022 to discover issues and identify potential security vulnerabilities in Tora Inu 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 Tora Inu 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 grieving 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	Tuesday Oct 11 2022 11:15:45 GMT+0000 (Coordinated Universal Time)
Finished	Wednesday Oct 12 2022 08:32:37 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	TORA.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

[illegible]

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 121

## low SEVERITY

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

## Source File

- TORA.sol

## Locations

```
120 function add(uint256 a, uint256 b) internal pure returns (uint256) {  
121     uint256 c = a + b;  
122     require(c >= a, "SafeMath: addition overflow");  
123  
124     return c;  
125 }
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 153

## low SEVERITY

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

## Source File

- TORA.sol

## Locations

```
152   require(b <= a, errorMessage);  
153   uint256 c = a - b;  
154  
155   return c;  
156   }  
157
```

# SWC-101 | ARITHMETIC OPERATION "\*" DISCOVERED

LINE 176

## low SEVERITY

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

## Source File

- TORA.sol

## Locations

```
175
176  uint256 c = a * b;
177  require(c / a == b, "SafeMath: multiplication overflow");
178
179  return c;
180
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 177

## low SEVERITY

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

## Source File

- TORA.sol

## Locations

```
176  uint256 c = a * b;  
177  require(c / a == b, "SafeMath: multiplication overflow");  
178  
179  return c;  
180  }  
181
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 212

## low SEVERITY

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

## Source File

- TORA.sol

## Locations

```
211     require(b > 0, errorMessage);
212     uint256 c = a / b;
213     // assert(a == b * c + a % b); // There is no case in which this doesn't hold
214
215     return c;
216
```



# SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 248

## low SEVERITY

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

## Source File

- TORA.sol

## Locations

```
247     require(b != 0, errorMessage);
248     return a % b;
249 }
250 }
251
252
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 475

## low SEVERITY

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

## Source File

- TORA.sol

## Locations

```
474  _owner = address(0);  
475  _lockTime = now + time;  
476  emit OwnershipTransferred(_owner, address(0));  
477  }  
478  
479
```

# SWC-101 | ARITHMETIC OPERATION "\*" DISCOVERED

LINE 535

## low SEVERITY

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

## Source File

- TORA.sol

## Locations

```
534 uint256 private constant MAX = ~uint256(0);
535 uint256 private _tTotal = 1_000_000_000 * 10**18;
536 uint256 private _rTotal = (MAX - (MAX % _tTotal));
537 uint256 private _tFeeTotal;
538
539
```

# SWC-101 | ARITHMETIC OPERATION "\*\*" DISCOVERED

LINE 535

## low SEVERITY

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

## Source File

- TORA.sol

## Locations

```
534 uint256 private constant MAX = ~uint256(0);  
535 uint256 private _tTotal = 1_000_000_000 * 10**18;  
536 uint256 private _rTotal = (MAX - (MAX % _tTotal));  
537 uint256 private _tFeeTotal;  
538  
539
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 536

## low SEVERITY

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

## Source File

- TORA.sol

## Locations

```
535  uint256 private _tTotal = 1_000_000_000 * 10**18;
536  uint256 private _rTotal = (MAX - (MAX % _tTotal));
537  uint256 private _tFeeTotal;
538
539  uint256 private _tBurnTotal;
540
```

# SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 536

## low SEVERITY

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

## Source File

- TORA.sol

## Locations

```
535 uint256 private _tTotal = 1_000_000_000 * 10**18;
536 uint256 private _rTotal = (MAX - (MAX % _tTotal));
537 uint256 private _tFeeTotal;
538
539 uint256 private _tBurnTotal;
540
```

# SWC-101 | ARITHMETIC OPERATION "\*" DISCOVERED

LINE 564

## low SEVERITY

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

## Source File

- TORA.sol

## Locations

```
563
564  uint256 public _maxTxAmount = 10_000_000 * 10**18;
565  uint256 public swapTokensAtAmount = 50_000_000 * 10**18;
566
567  event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
568
```

# SWC-101 | ARITHMETIC OPERATION "\*\*" DISCOVERED

LINE 564

## low SEVERITY

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

## Source File

- TORA.sol

## Locations

```
563
564  uint256 public _maxTxAmount = 10_000_000 * 10**18;
565  uint256 public swapTokensAtAmount = 50_000_000 * 10**18;
566
567  event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
568
```



# SWC-101 | ARITHMETIC OPERATION "\*" DISCOVERED

LINE 565

## low SEVERITY

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

## Source File

- TORA.sol

## Locations

```
564  uint256 public _maxTxAmount = 10_000_000 * 10**18;
565  uint256 public swapTokensAtAmount = 50_000_000 * 10**18;
566
567  event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
568  event SwapAndLiquifyEnabledUpdated(bool enabled);
569
```

# SWC-101 | ARITHMETIC OPERATION "\*\*" DISCOVERED

LINE 565

## low SEVERITY

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

## Source File

- TORA.sol

## Locations

```
564 uint256 public _maxTxAmount = 10_000_000 * 10**18;
565 uint256 public swapTokensAtAmount = 50_000_000 * 10**18;
566
567 event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
568 event SwapAndLiquifyEnabledUpdated(bool enabled);
569
```

# SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 698

## low SEVERITY

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

## Source File

- TORA.sol

## Locations

```
697   require(!_isExcluded[account], "Account is already excluded");
698   for (uint256 i = 0; i < _excluded.length; i++) {
699       if (_excluded[i] == account) {
700           _excluded[i] = _excluded[_excluded.length - 1];
701           _tOwned[account] = 0;
702       }
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 700

## low SEVERITY

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

## Source File

- TORA.sol

## Locations

```
699   if (_excluded[i] == account) {  
700     _excluded[i] = _excluded[_excluded.length - 1];  
701     _tOwned[account] = 0;  
702     _isExcluded[account] = false;  
703     _excluded.pop();  
704   }
```

# SWC-101 | ARITHMETIC OPERATION "\*\*" DISCOVERED

LINE 753

## low SEVERITY

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

## Source File

- TORA.sol

## Locations

```
752     _maxTxAmount = _tTotal.mul(maxTxPercent).div(  
753     10**2  
754     );  
755 }  
756  
757
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 784

## low SEVERITY

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

## Source File

- TORA.sol

## Locations

```
783
784  uint256 extrafee = tburnFee + tmarketingFee;
785
786  uint256 tTransferAmount = tAmount.sub(tFee).sub(tLiquidity).sub(extrafee);
787  return (tTransferAmount, tFee, tLiquidity);
788
```

# SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 815

## low SEVERITY

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

## Source File

-TORA.sol

## Locations

```
814  uint256 tSupply = _tTotal;
815  for (uint256 i = 0; i < _excluded.length; i++) {
816    if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return
      (_rTotal, _tTotal);
817    rSupply = rSupply.sub(_rOwned[_excluded[i]]);
818    tSupply = tSupply.sub(_tOwned[_excluded[i]]);
819  }
```

# SWC-101 | ARITHMETIC OPERATION "\*\*" DISCOVERED

LINE 836

## low SEVERITY

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

## Source File

- TORA.sol

## Locations

```
835     return _amount.mul(_taxFee).div(  
836         10**2  
837     );  
838 }  
839  
840
```



# SWC-101 | ARITHMETIC OPERATION "\*\*" DISCOVERED

LINE 842

## low SEVERITY

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

## Source File

- TORA.sol

## Locations

```
841     return _amount.mul(_liquidityFee).div(  
842         10**2  
843     );  
844 }  
845  
846
```

# SWC-101 | ARITHMETIC OPERATION "\*\*" DISCOVERED

LINE 848

## low SEVERITY

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

## Source File

- TORA.sol

## Locations

```
847     return _amount.mul(_burnFee).div(  
848         10**2  
849     );  
850 }  
851  
852
```

# SWC-101 | ARITHMETIC OPERATION "\*\*" DISCOVERED

LINE 854

## low SEVERITY

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

## Source File

- TORA.sol

## Locations

```
853     return _amount.mul(_marketingFee).div(  
854         10**2  
855     );  
856 }  
857  
858
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 909

## low SEVERITY

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

## Source File

- TORA.sol

## Locations

```
908
909  uint256 contractTokenBalanceforLP = balanceOf(address(this)) -
    _pendingMarketingFee;
910
911  bool canSwap = contractTokenBalanceforLP >= swapTokensAtAmount;
912  if (
913
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 964

## low SEVERITY

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

## Source File

- TORA.sol

## Locations

```
963     function swapOnDemand() external onlyOwner {
964         uint256 contractTokenBalanceforlp = balanceOf(address(this)) -
        _pendingMarketingFee;
965         if (contractTokenBalanceforlp>0) {
966             inSwapAndLiquify = true;
967             swapAndLiquify(contractTokenBalanceforlp);
968         }
```

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

LINE 700

## low SEVERITY

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

## Source File

- TORA.sol

## Locations

```
699   if (_excluded[i] == account) {  
700     _excluded[i] = _excluded[_excluded.length - 1];  
701     _tOwned[account] = 0;  
702     _isExcluded[account] = false;  
703     _excluded.pop();  
704
```

## SWC-103 | A FLOATING PRAGMA IS SET.

LINE 22

### 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

- TORA.sol

### Locations

```
21
22  pragma solidity ^0.6.12;
23  // SPDX-License-Identifier: Unlicensed
24  interface IERC20 {
25
26
```

## SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 561

### 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

- TORA.sol

### Locations

```
560
561  bool inSwapAndLiquify;
562  bool public swapAndLiquifyEnabled = true;
563
564  uint256 public _maxTxAmount = 10_000_000 * 10**18;
565
```



## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 699

### low SEVERITY

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

### Source File

-TORA.sol

### Locations

```
698   for (uint256 i = 0; i < _excluded.length; i++) {  
699     if (_excluded[i] == account) {  
700       _excluded[i] = _excluded[_excluded.length - 1];  
701       _tOwned[account] = 0;  
702       _isExcluded[account] = false;  
703     }
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 700

### low SEVERITY

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

### Source File

- TORA.sol

### Locations

```
699   if (_excluded[i] == account) {  
700     _excluded[i] = _excluded[_excluded.length - 1];  
701     _tOwned[account] = 0;  
702     _isExcluded[account] = false;  
703     _excluded.pop();  
704   }
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 700

### low SEVERITY

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

### Source File

- TORA.sol

### Locations

```
699   if (_excluded[i] == account) {  
700     _excluded[i] = _excluded[_excluded.length - 1];  
701     _tOwned[account] = 0;  
702     _isExcluded[account] = false;  
703     _excluded.pop();  
704   }
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 816

### low SEVERITY

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

### Source File

-TORA.sol

### Locations

```
815   for (uint256 i = 0; i < _excluded.length; i++) {  
816     if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return  
      (_rTotal, _tTotal);  
817     rSupply = rSupply.sub(_rOwned[_excluded[i]]);  
818     tSupply = tSupply.sub(_tOwned[_excluded[i]]);  
819   }  
820
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 816

### low SEVERITY

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

### Source File

-TORA.sol

### Locations

```
815   for (uint256 i = 0; i < _excluded.length; i++) {  
816     if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return  
      (_rTotal, _tTotal);  
817     rSupply = rSupply.sub(_rOwned[_excluded[i]]);  
818     tSupply = tSupply.sub(_tOwned[_excluded[i]]);  
819   }  
820
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 817

### low SEVERITY

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

### Source File

-TORA.sol

### Locations

```
816   if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return
      (_rTotal, _tTotal);
817   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
818   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
819   }
820   if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
821
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 818

### low SEVERITY

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

### Source File

- TORA.sol

### Locations

```
817   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
818   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
819   }
820   if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
821   return (rSupply, tSupply);
822
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1002

### low SEVERITY

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

### Source File

- TORA.sol

### Locations

```
1001     address[] memory path = new address[](2);
1002     path[0] = address(this);
1003     path[1] = uniswapV2Router.WETH();
1004
1005     _approve(address(this), address(uniswapV2Router), tokenAmount);
1006
```



## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1003

### low SEVERITY

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

### Source File

- TORA.sol

### Locations

```
1002  path[0] = address(this);
1003  path[1] = uniswapV2Router.WETH();
1004
1005  _approve(address(this), address(uniswapV2Router), tokenAmount);
1006
1007
```

# 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.