QuillAudits

Audit Report
August, 2021

TripLeverage

Contents

Scope of Audit

The scope of this audit was to analyze and document the TripLeverageToken smart contract codebase for quality, security, and correctness.

Checked Vulnerabilities

We have scanned the smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis
In this step we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.
SmartCheck.

Static Analysis
Static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step a series of automated tools are used to test security of smart contracts.

Code Review / Manual Analysis
Manual Analysis or review of code was done to identify new vulnerability or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of automated analysis were manually verified.

Gas Consumption
In this step we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit
Remix IDE, Truffle, Truffle Team, Ganache, Solhint, Mythril, Slither, SmartCheck.

Issue Categories

Every issue in this report has been assigned with a severity level. There are four levels of severity and each of them has been explained below.

High severity issues

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality and we recommend these issues to be fixed before moving to a live environment.

Medium level severity issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems and they should still be fixed.

Low level severity issues

Low level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are severity four issues which indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Number of issues per severity

Type	High	Medium	Low	Informational
Open	0	0	0	0
Acknowledged	0	0	0	0
Closed	0	1	3	2

Introduction

During the period of **July 26, 2021 to August 6, 2021** - QuillAudits Team performed a security audit for TripLeverageToken smart contracts. The code for the audit was provided by TripLeverageToken and their hashes are mentioned below:

Smart Contract Online Code: https://testnet.bscscan.com/address/0xf94c27e569616ce9fed105d65063d32a9cb6b2cd#code

MD5 hash: 5169305861627A57E15784C2F7D1A8E6

Updated Smart Contract Online Code: https://testnet.bscscan.com/address/0xd8d1b4A16387bF2e16cFB44277811650B397aDF5#code

Updated MD5 hash: 7EF06FA37660453D4A5EB3B2027F5301

Updated Smart Contract Online Code: https://testnet.bscscan.com/address/0x48Db027D16b3395698E989bf9d874b06b1187844#code

Updated MD5 hash: C089210B999F8942F747E5D1E3C01EA1

Updated Smart Contract Online Code: https://testnet.bscscan.com/address/0x9CBba68B33d8b51cE94DA0E90BCE7659C552dbCF#code

Updated MD5 hash: 15C5729E8DF262FA814D695DAA8E22E6

Updated Smart Contract Online Code: https://testnet.bscscan.com/address/0xC0F71F9dD27811BB945C11B51cccc2AA3f92a36B#code

Updated MD5 hash: 6C34C20ABF6F61F0459E50458EB22D6A

Issues Found – Code Review / Manual Testing

High severity issues

No issues were found

Medium severity issues

1. **Division before multiplication**

   Description
   Solidity being resource constraint language, dividing any amount and then multiplying will cause a discrepancy in the outcome. Therefore always multiply the amount first and then divide it.

   Remediation
   Consider ordering multiplication before division.

   Status: Fixed

Low level severity issues

2. Missing zero address validation

Line	Code

418	constructor (address _charity) {
	charityAddress = _charity;
	_rOwned[_msgSender()] = _rTotal;
	emit Transfer(address(0), _msgSender(), _tTotal);
	
	_isExcluded[_charity] = true;
	_excluded.push(_charity);
	}

Description

Detecting missing zero address validation to prevent contract losses, in function constructor and setCharitywallet check the address is not zero__charity.

Remediation

Check that the address is not zero.

Status: Fixed

3. Infinite loop possibility

Line	Code

526	function includeAccount(address account) external onlyOwner() {
	require(_isExcluded[account], "Account is already excluded");
	for (uint256 i = 0; i < _excluded.length; i++) {
	if (_excluded[i] == account) {
	_excluded[i] = _excluded[_excluded.length - 1];
	_tOwned[account] = 0;
	_isExcluded[account] = false;
	_excluded.pop();
	break;
	}
	}
	}

Description
If there are so many excluded accounts, then this logic will fail, as it might hit the block's gas limit. If there are very limited excluded accounts, then this will work, but will cost more gas.

Status: Fixed

4. Document specification has not been done

Description
The burn is automatic (2%), and the total max supply is updated accordingly, which can be seen from the blockchain scans such as bscscan.

Status: Fixed

Informational

5. Remove unused comment and code

Line	Code
487	// function totalFees() public view returns (uint256) { // return _tFeeTotal; // }

Description
Remove this code in the comment if not used.

Remediation
Remove this code in the comment if not used.

Status: Fixed

6. Make variables constant

Line	Code
414	string private _name = 'Trip Leverage Token'; string private _symbol = 'TLT'; uint8 private _decimals = 9;

Description

_name, _symbol, _decimals variable values will be unchanged.

Remediation

Please add the "constant" keyword.

Status: Fixed

Functional test

Function Names	Testing results
name	Passed
symbol	Passed
decimals	Passed
totalSupply	Passed
balanceOf	Passed
transfer	Passed
allowance	Passed
approve	Passed
transferFrom	Passed
increaseAllowance	Passed
decreaseAllowance	Passed
isExcluded	Passed
setCharitywallet	Passed
reflect	Passed
reflectionFromToken	Passed
tokenFromReflection	Passed
excludeAccount	Passed
includeAccount	Passed
_approve	Passed
_transfer	Passed

Function Names	Testing results
_transferStandard	Passed
_transferToExcluded	Passed
_transferFromExcluded	Passed
_transferBothExcluded	Passed
_reflectFee	Passed
_getValues	Passed
_getTValues	Passed
_getRValues	Passed
_getRate	Passed
_getCurrentSupply	Passed
owner	Passed
onlyOwner	Passed
renounceOwnership	Passed
transferOwnership	Passed
_msgSender	Passed
_msgData	Passed

Automated Testing

Slither

```
INFO:Detectors:
TripLeverageToken._getTValues(uint256) (TripLeverageToken.sol#619-626) performs a multiplication on the result of a division:
        -tTax = tAmount.div(100).mul(3) (TripLeverageToken.sol#620)
TripLeverageToken._getTValues(uint256) (TripLeverageToken.sol#619-626) performs a multiplication on the result of a division:
        -tBurnValue = tAmount.div(100).mul(2) (TripLeverageToken.sol#621)
TripLeverageToken._getTValues(uint256) (TripLeverageToken.sol#619-626) performs a multiplication on the result of a division:
        -tCharity = tAmount.div(100).mul(5) (TripLeverageToken.sol#622)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
TripLeverageToken.allowance(address,address).owner (TripLeverageToken.sol#453) shadows:
        - Ownable.owner() (TripLeverageToken.sol#361-363) (function)
TripLeverageToken._approve(address,address,uint256).owner (TripLeverageToken.sol#539) shadows:
        - Ownable.owner() (TripLeverageToken.sol#361-363) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Ownable.constructor().msgSender (TripLeverageToken.sol#354) lacks a zero-check on :
                - _owner = msgSender (TripLeverageToken.sol#355)
TripLeverageToken.constructor(address)._charity (TripLeverageToken.sol#418) lacks a zero-check on :
                - charityAddress = _charity (TripLeverageToken.sol#419)
TripLeverageToken.setCharitywallet(address)._charity (TripLeverageToken.sol#482) lacks a zero-check on :
                - charityAddress = _charity (TripLeverageToken.sol#483)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Address.isContract(address) (TripLeverageToken.sol#234-243) uses assembly
        - INLINE ASM (TripLeverageToken.sol#241)
Address._functionCallWithValue(address,bytes,uint256,string) (TripLeverageToken.sol#322-343) uses assembly
        - INLINE ASM (TripLeverageToken.sol#335-338)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address._functionCallWithValue(address,bytes,uint256,string) (TripLeverageToken.sol#322-343) is never used and should be removed
Address.functionCall(address,bytes) (TripLeverageToken.sol#285-287) is never used and should be removed
Address.functionCall(address,bytes,string) (TripLeverageToken.sol#294-296) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (TripLeverageToken.sol#308-310) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (TripLeverageToken.sol#317-320) is never used and should be removed
Address.isContract(address) (TripLeverageToken.sol#234-243) is never used and should be removed
Address.sendValue(address,uint256) (TripLeverageToken.sol#260-266) is never used and should be removed
Context._msgData() (TripLeverageToken.sol#14-17) is never used and should be removed
SafeMath.mod(uint256,uint256) (TripLeverageToken.sol#195-197) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (TripLeverageToken.sol#210-213) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (TripLeverageToken.sol#7) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (TripLeverageToken.sol#260-266):
        - (success) = recipient.call{value: amount}() (TripLeverageToken.sol#264)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (TripLeverageToken.sol#322-343):
        - (success,returndata) = target.call{value: weiValue}(data) (TripLeverageToken.sol#326)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Event TripLeverageTokencharityWallet(address,uint256) (TripLeverageToken.sol#402) is not in CapWords
Parameter TripLeverageToken.setCharitywallet(address)._charity (TripLeverageToken.sol#482) is not in mixedCase
Function TripLeverageToken._getRate() (TripLeverageToken.sol#637-640) is not in mixedCase
Function TripLeverageToken._getCurrentSupply() (TripLeverageToken.sol#642-652) is not in mixedCase
Constant TripLeverageToken._tTotal (TripLeverageToken.sol#408) is not in UPPER_CASE_WITH_UNDERSCORES
Variable TripLeverageToken._rTotal (TripLeverageToken.sol#409) is not in mixedCase
Variable TripLeverageToken.TokenBurns (TripLeverageToken.sol#412) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (TripLeverageToken.sol#15)" inContext (TripLeverageToken.sol#9-18)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Variable TripLeverageToken._transferFromExcluded(address,address,uint256).rTransferAmount (TripLeverageToken.sol#584) is too similar to T
ripLeverageToken._transferFromExcluded(address,address,uint256).tTransferAmount (TripLeverageToken.sol#584)
Variable TripLeverageToken._transferToExcluded(address,address,uint256).rTransferAmount (TripLeverageToken.sol#574) is too similar to Tri
pLeverageToken._transferFromExcluded(address,address,uint256).tTransferAmount (TripLeverageToken.sol#584)
Variable TripLeverageToken._getRValues(uint256,uint256,uint256,uint256,uint256).rTransferAmount (TripLeverageToken.sol#633) is too simila
r to TripLeverageToken._transferBothExcluded(address,address,uint256).tTransferAmount (TripLeverageToken.sol#594)
Variable TripLeverageToken._transferFromExcluded(address,address,uint256).rTransferAmount (TripLeverageToken.sol#584) is too similar to T
ripLeverageToken._transferToExcluded(address,address,uint256).tTransferAmount (TripLeverageToken.sol#574)
```

```
INFO:Detectors:
Variable TripLeverageToken._transferFromExcluded(address,address,uint256).rTransferAmount (TripLeverageToken.sol#584) is too similar to T
ripLeverageToken._transferFromExcluded(address,address,uint256).tTransferAmount (TripLeverageToken.sol#584)
Variable TripLeverageToken._transferToExcluded(address,address,uint256).rTransferAmount (TripLeverageToken.sol#574) is too similar to Tri
pLeverageToken._transferFromExcluded(address,address,uint256).tTransferAmount (TripLeverageToken.sol#584)
Variable TripLeverageToken._getRValues(uint256,uint256,uint256,uint256,uint256).rTransferAmount (TripLeverageToken.sol#633) is too simila
r to TripLeverageToken._transferBothExcluded(address,address,uint256).tTransferAmount (TripLeverageToken.sol#594)
Variable TripLeverageToken._transferFromExcluded(address,address,uint256).rTransferAmount (TripLeverageToken.sol#584) is too similar to T
ripLeverageToken._transferToExcluded(address,address,uint256).tTransferAmount (TripLeverageToken.sol#574)
Variable TripLeverageToken._transferToExcluded(address,address,uint256).rTransferAmount (TripLeverageToken.sol#574) is too similar to Tri
pLeverageToken._transferToExcluded(address,address,uint256).tTransferAmount (TripLeverageToken.sol#574)
Variable TripLeverageToken._transferStandard(address,address,uint256).rTransferAmount (TripLeverageToken.sol#565) is too similar to TripL
everageToken._transferBothExcluded(address,address,uint256).tTransferAmount (TripLeverageToken.sol#594)
Variable TripLeverageToken._getRValues(uint256,uint256,uint256,uint256,uint256).rTransferAmount (TripLeverageToken.sol#633) is too simila
r to TripLeverageToken._getTValues(uint256).tTransferAmount (TripLeverageToken.sol#623)
Variable TripLeverageToken._transferStandard(address,address,uint256).rTransferAmount (TripLeverageToken.sol#565) is too similar to TripL
everageToken._getTValues(uint256).tTransferAmount (TripLeverageToken.sol#623)
Variable TripLeverageToken._transferStandard(address,address,uint256).rTransferAmount (TripLeverageToken.sol#565) is too similar to TripL
everageToken._transferToExcluded(address,address,uint256).tTransferAmount (TripLeverageToken.sol#574)
Variable TripLeverageToken._transferBothExcluded(address,address,uint256).rTransferAmount (TripLeverageToken.sol#594) is too similar to T
ripLeverageToken._transferFromExcluded(address,address,uint256).tTransferAmount (TripLeverageToken.sol#584)
Variable TripLeverageToken._getRValues(uint256,uint256,uint256,uint256,uint256).rTransferAmount (TripLeverageToken.sol#633) is too simila
r to TripLeverageToken._transferFromExcluded(address,address,uint256).tTransferAmount (TripLeverageToken.sol#584)
Variable TripLeverageToken.reflectionFromToken(uint256,bool).rTransferAmount (TripLeverageToken.sol#506) is too similar to TripLeverageTo
ken._transferFromExcluded(address,address,uint256).tTransferAmount (TripLeverageToken.sol#584)
Variable TripLeverageToken._transferStandard(address,address,uint256).rTransferAmount (TripLeverageToken.sol#565) is too similar to TripL
everageToken._transferFromExcluded(address,address,uint256).tTransferAmount (TripLeverageToken.sol#584)
Variable TripLeverageToken._getRValues(uint256,uint256,uint256,uint256,uint256).rTransferAmount (TripLeverageToken.sol#633) is too simila
r to TripLeverageToken._transferToExcluded(address,address,uint256).tTransferAmount (TripLeverageToken.sol#574)
Variable TripLeverageToken.reflectionFromToken(uint256,bool).rTransferAmount (TripLeverageToken.sol#506) is too similar to TripLeverageTo
ken._transferToExcluded(address,address,uint256).tTransferAmount (TripLeverageToken.sol#574)
Variable TripLeverageToken.reflectionFromToken(uint256,bool).rTransferAmount (TripLeverageToken.sol#506) is too similar to TripLeverageTo
ken._getTValues(uint256).tTransferAmount (TripLeverageToken.sol#623)
Variable TripLeverageToken._transferToExcluded(address,address,uint256).rTransferAmount (TripLeverageToken.sol#574) is too similar to Tri
pLeverageToken._transferStandard(address,address,uint256).tTransferAmount (TripLeverageToken.sol#565)
Variable TripLeverageToken._transferFromExcluded(address,address,uint256).rTransferAmount (TripLeverageToken.sol#584) is too similar to T
ripLeverageToken._transferBothExcluded(address,address,uint256).tTransferAmount (TripLeverageToken.sol#594)
Variable TripLeverageToken._transferToExcluded(address,address,uint256).rTransferAmount (TripLeverageToken.sol#574) is too similar to Tri
Variable TripLeverageToken.reflectionFromToken(uint256,bool).rTransferAmount (TripLeverageToken.sol#506) is too similar to TripLeverageTo
ken._transferStandard(address,address,uint256).tTransferAmount (TripLeverageToken.sol#565)
Variable TripLeverageToken._transferFromExcluded(address,address,uint256).rTransferAmount (TripLeverageToken.sol#584) is too similar to T
ripLeverageToken._transferStandard(address,address,uint256).tTransferAmount (TripLeverageToken.sol#565)
Variable TripLeverageToken._transferBothExcluded(address,address,uint256).rTransferAmount (TripLeverageToken.sol#594) is too similar to T
ripLeverageToken._getTValues(uint256).tTransferAmount (TripLeverageToken.sol#623)
Variable TripLeverageToken._transferToExcluded(address,address,uint256).rTransferAmount (TripLeverageToken.sol#574) is too similar to Tri
pLeverageToken._transferBothExcluded(address,address,uint256).tTransferAmount (TripLeverageToken.sol#594)
Variable TripLeverageToken._transferBothExcluded(address,address,uint256).rTransferAmount (TripLeverageToken.sol#594) is too similar to T
ripLeverageToken._transferToExcluded(address,address,uint256).tTransferAmount (TripLeverageToken.sol#574)
Variable TripLeverageToken.reflectionFromToken(uint256,bool).rTransferAmount (TripLeverageToken.sol#506) is too similar to TripLeverageTo
ken._transferBothExcluded(address,address,uint256).tTransferAmount (TripLeverageToken.sol#594)
Variable TripLeverageToken._transferBothExcluded(address,address,uint256).rTransferAmount (TripLeverageToken.sol#594) is too similar to T
ripLeverageToken._transferBothExcluded(address,address,uint256).tTransferAmount (TripLeverageToken.sol#594)
Variable TripLeverageToken._transferBothExcluded(address,address,uint256).rTransferAmount (TripLeverageToken.sol#594) is too similar to T
ripLeverageToken._transferStandard(address,address,uint256).tTransferAmount (TripLeverageToken.sol#565)
Variable TripLeverageToken._transferFromExcluded(address,address,uint256).rTransferAmount (TripLeverageToken.sol#584) is too similar to T
ripLeverageToken._getTValues(uint256).tTransferAmount (TripLeverageToken.sol#623)
Variable TripLeverageToken._transferStandard(address,address,uint256).rTransferAmount (TripLeverageToken.sol#565) is too similar to TripL
everageToken._transferStandard(address,address,uint256).tTransferAmount (TripLeverageToken.sol#565)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
TripLeverageToken.slitherConstructorConstantVariables() (TripLeverageToken.sol#393-654) uses literals with too many digits:
        - _tTotal = 120000000 * 10 ** 9 (TripLeverageToken.sol#408)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
TripLeverageToken._decimals (TripLeverageToken.sol#416) should be constant
TripLeverageToken._name (TripLeverageToken.sol#414) should be constant
TripLeverageToken._symbol (TripLeverageToken.sol#415) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
owner() should be declared external:
        - Ownable.owner() (TripLeverageToken.sol#361-363)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (TripLeverageToken.sol#378-381)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (TripLeverageToken.sol#386-390)
name() should be declared external:
```

```
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (TripLeverageToken.sol#378-381)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (TripLeverageToken.sol#386-390)
name() should be declared external:
        - TripLeverageToken.name() (TripLeverageToken.sol#427-429)
symbol() should be declared external:
        - TripLeverageToken.symbol() (TripLeverageToken.sol#431-433)
decimals() should be declared external:
        - TripLeverageToken.decimals() (TripLeverageToken.sol#435-437)
totalSupply() should be declared external:
        - TripLeverageToken.totalSupply() (TripLeverageToken.sol#439-441)
balanceOf(address) should be declared external:
        - TripLeverageToken.balanceOf(address) (TripLeverageToken.sol#443-446)
transfer(address,uint256) should be declared external:
        - TripLeverageToken.transfer(address,uint256) (TripLeverageToken.sol#448-451)
allowance(address,address) should be declared external:
        - TripLeverageToken.allowance(address,address) (TripLeverageToken.sol#453-455)
approve(address,uint256) should be declared external:
        - TripLeverageToken.approve(address,uint256) (TripLeverageToken.sol#457-460)
transferFrom(address,address,uint256) should be declared external:
        - TripLeverageToken.transferFrom(address,address,uint256) (TripLeverageToken.sol#462-466)
increaseAllowance(address,uint256) should be declared external:
        - TripLeverageToken.increaseAllowance(address,uint256) (TripLeverageToken.sol#468-471)
decreaseAllowance(address,uint256) should be declared external:
        - TripLeverageToken.decreaseAllowance(address,uint256) (TripLeverageToken.sol#473-476)
isExcluded(address) should be declared external:
        - TripLeverageToken.isExcluded(address) (TripLeverageToken.sol#478-480)
setCharitywallet(address) should be declared external:
        - TripLeverageToken.setCharitywallet(address) (TripLeverageToken.sol#482-485)
reflect(uint256) should be declared external:
        - TripLeverageToken.reflect(uint256) (TripLeverageToken.sol#491-498)
reflectionFromToken(uint256,bool) should be declared external:
        - TripLeverageToken.reflectionFromToken(uint256,bool) (TripLeverageToken.sol#500-509)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:TripLeverageToken.sol analyzed (6 contracts with 75 detectors), 84 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

SOLIDITY STATIC ANALYSIS

SOLIDITY STATIC ANALYSIS

contracts/TripLeverageToken.sol

Security

Transaction origin:

INTERNAL ERROR in module Transaction origin: can't convert undefined to object

Pos: not available

Check-effects-interaction:

INTERNAL ERROR in module Check-effects-interaction: can't convert undefined to object

Pos: not available

Inline assembly:

INTERNAL ERROR in module Inline assembly: can't convert undefined to object

Pos: not available

Block timestamp:

INTERNAL ERROR in module Block timestamp: can't convert undefined to object

Pos: not available

Low level calls:

INTERNAL ERROR in module Low level calls: can't convert undefined to object

Pos: not available

Selfdestruct:

INTERNAL ERROR in module Selfdestruct: can't convert undefined to object
Pos: not available

Gas & Economy

This on local calls:

INTERNAL ERROR in module This on local calls: can't convert undefined to object
Pos: not available

Delete dynamic array:

INTERNAL ERROR in module Delete dynamic array: can't convert undefined to object
Pos: not available

For loop over dynamic array:

INTERNAL ERROR in module For loop over dynamic array: can't convert undefined to object
Pos: not available

Ether transfer in loop:

INTERNAL ERROR in module Ether transfer in loop: can't convert undefined to object
Pos: not available

ERC

ERC20:

INTERNAL ERROR in module ERC20: can't convert undefined to object
Pos: not available

Miscellaneous

Constant/View/Pure functions:

INTERNAL ERROR in module Constant/View/Pure functions: can't convert undefined to object
Pos: not available

Similar variable names:

INTERNAL ERROR in module Similar variable names: can't convert undefined to object
Pos: not available

No return:

INTERNAL ERROR in module No return: can't convert undefined to object
Pos: not available

Guard conditions:

INTERNAL ERROR in module Guard conditions: can't convert undefined to object
Pos: not available

String length:

INTERNAL ERROR in module String length: can't convert undefined to object
Pos: not available

SOLHINT LINTER

```
contracts/TripLeverageToken.sol:7:1: Error: Compiler version ^0.8.0 does not satisfy the r
semver requirement

contracts/TripLeverageToken.sol:353:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)

contracts/TripLeverageToken.sol:401:28: Error: Variable name must be in mixedCase

contracts/TripLeverageToken.sol:402:5: Error: Event name must be in CamelCase

contracts/TripLeverageToken.sol:408:30: Error: Constant name must be in capitalized
SNAKE_CASE

contracts/TripLeverageToken.sol:412:20: Error: Variable name must be in mixedCase

contracts/TripLeverageToken.sol:414:28: Error: Use double quotes for string literals

contracts/TripLeverageToken.sol:415:30: Error: Use double quotes for string literals

contracts/TripLeverageToken.sol:418:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
```

Results

No major issues were found. Some false positive errors were reported by the tool. All the other issues have been categorized above according to their level of severity.

Closing Summary

In this report, we have considered the security of the TripLeverageToken platform. We performed our audit according to the procedure described above.

The majority of the concerns addressed above have been acknowledged, implemented and verified.

Disclaimer

Quillhash audit is not a security warranty, investment advice, or an endorsement of the TripLeverageToken platform. This audit does not provide a security or correctness guarantee of the audited smart contracts. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the TripLeverageToken Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

TripLeverage

QuillAudits

Canada, India, Singapore and United Kingdom

audits.quillhash.com

audits@quillhash.com