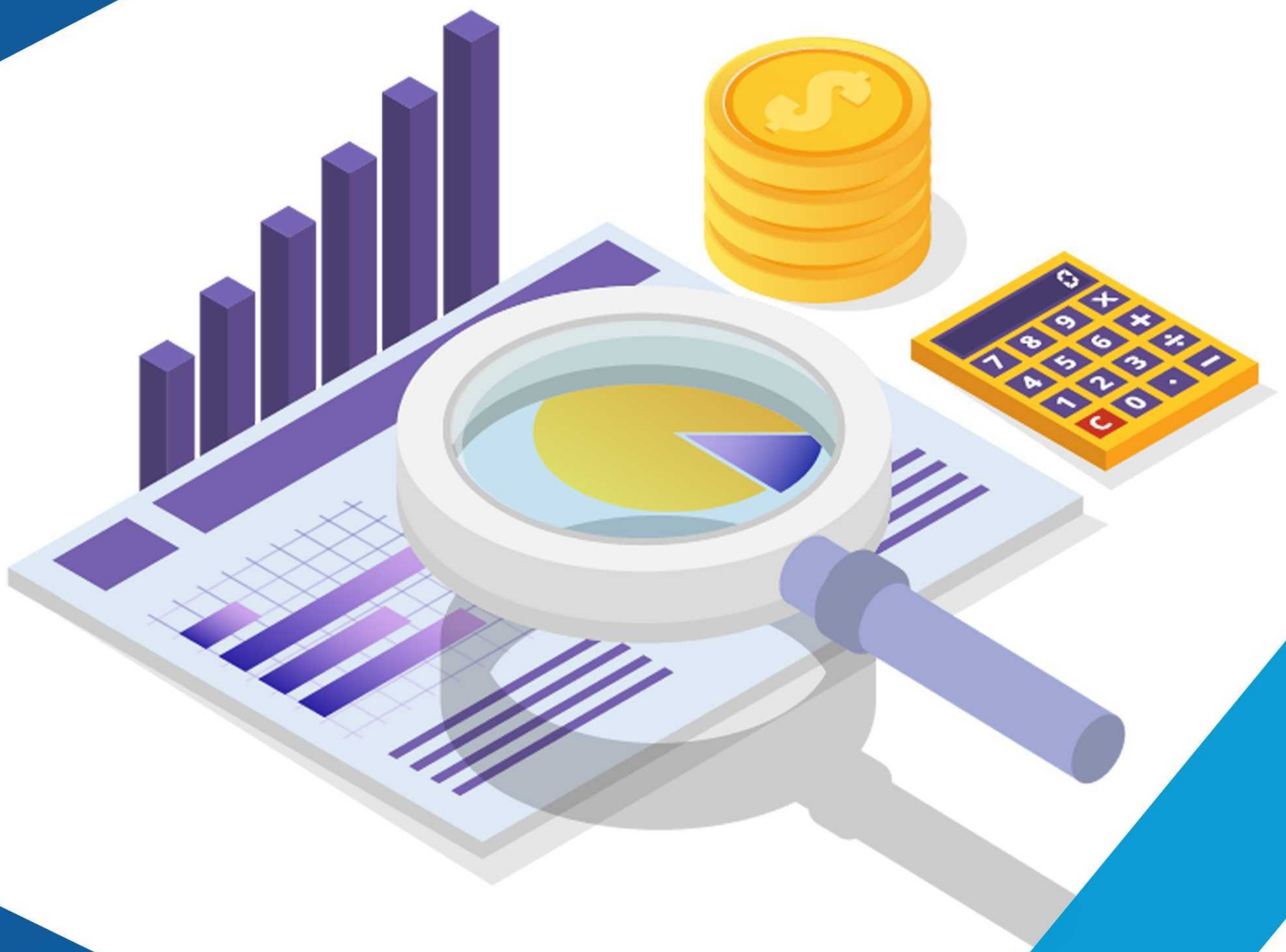
QuillAudits



Audit Report
June, 2021

LoLz FINANCE

Contents

Scope of Audit

The scope of this audit was to analyze and document the LOLZTOKEN Token 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	1
Acknowledged	0	1	1	7
Closed	0	1	4	3

Introduction

During the period of **June 21, 2021 to June 25, 2021** - QuillAudits Team performed a security audit for LOLZTOKEN smart contracts.

The code for the audit was taken from following the official link:
https://github.com/LoLzFinance1/LOLZToken
Branch: audit
Commit hash: 53bd9002b82b5fa80024a3d3ad224945a880b306

The following code was updated with remediations:
https://github.com/LoLzFinance1/LOLZToken
Branch: audit_fixes
Commit hash: 6968a1848895494c0cb0376b77c4c88423c2d107

Issues Found – Code Review / Manual Testing

High severity issues

No issues were found.

Medium severity issues

1. Centralization Risks

Description
The role owner has the authority to

- update settings
- manage the list containing contracts excluding from reward, fee, or max transaction limitation

Remediation
We advise the client to handle the governance account carefully to avoid any potential hack. We also advise the client to consider the following solutions:

- With reasonable latency for community awareness on privileged operations;
- Multisig with community-voted 3rd-party independent co-signers;
- DAO or Governance module increasing transparency and community involvement;

Status: Acknowledged by the auditee.
The owner will be set to 0 after the deployment.

2. Funds in this contract will be locked

Description
Contract with a payable function, but without a withdrawal capacity. Since **receive() external payable {}** is utilized without a withdrawal function. Thus, every Ether sent to this contract will be lost.

Remediation
Remove the payable attribute or add a withdrawal function for this contract.

Status: Fixed
The LOLZTOKEN team made some fixes based on our recommendations. We address below the fixes introduced up to commit d8ca4363f141aa84ce6830fc067499c4b6de37f9 of the repository, which now implements the withdrawal function for the token contract.

Low level severity issues

3. Redundant operation

Line	Code

374	else if (!_isExcluded[sender] && !_isExcluded[recipient]) {
	_transferStandard(sender, recipient, amount);
	}

Description
When the contract enters the branch **else if (!_isExcluded[sender] && !_isExcluded[recipient])** or **else**, the contract will execute the same piece of code **_transferStandard(sender, recipient, amount)**;

Remediation
We recommend removing the following code:

Line	Code

374	else if (!_isExcluded[sender] && !_isExcluded[recipient]) {
	_transferStandard(sender, recipient, amount);
	}

Status: Fixed
The redundant code has been removed.

4. Redundant variables

Line	Code
374	bool inSwapAndLiquify; bool public swapAndLiquifyEnabled;

Description
Unused state variables **inSwapAndLiquify** and **swapAndLiquifyEnabled** were found.

Remediation
We recommend removing those unused variables.

Status: Fixed

5. Missing Range Check for Input Variable

Line	Code

219	function setTaxFeePercent(uint256 taxFee) external onlyOwner() {
	_taxFee = taxFee;
	}

223	function setLiquidityFeePercent(uint256 liquidityFee) external onlyOwner()
	{
	_liquidityFee = liquidityFee;
	}

227	function setDevelopmentFeePercent(uint256 developmentFee) external
	onlyOwner() {
	_developmentFee = developmentFee;
	}

231	function setMaxTxPercent(uint256 maxTxPercent) external onlyOwner() {
	_maxTxAmount = _tTotal.mul(maxTxPercent).div(
	10**2
	);
	}

Description

The role can set the following state variables arbitrary large or small causing potential risks in fees and anti whale :

- _taxFee
- _liquidityFee
- _developmentFee
- _maxTxAmount

Remediation

We recommend setting ranges and check the following input variables:

- taxFee
- liquidityFee
- developmentFee
- maxTxPercent

Status: Fixed

A range check has been implemented for the aforementioned variables.

6. Incorrect Error Message

Line	Code
184	require(_isExcluded[account], "Account is already excluded");
158	require(tAmount <= _tTotal, "Amount must be less than supply");
169	require(rAmount <= _rTotal, "Amount must be less than total reflections");

Description
The error message in **require(_isExcluded[account], "Account is already excluded")** does not describe the error correctly.

The error message in **require(tAmount <= _tTotal, "Amount must be less than supply")** and **(rAmount <= _rTotal, "Amount must be less than total reflections")** do not describe the error accurately.

Remediation
We recommend changing **"Account is already excluded"** to "**Account is not excluded"**.

We recommend changing **"Amount must be less than supply"** to **"Amount must be less than or equal to supply"**.

We recommend changing **"Amount must be less than total reflections"** to **“Amount must be less than or equal to total reflections”.**

Status: Fixed
The message errors have been corrected for the require functions listed in the table above.

7. Missing zero address validation

Description

We've detected missing zero address validation for **liquidityAddress** and **developmentAddress** in the constructor.

```
constructor (address _liquidityAddress, address _developmentAddress) public {
        liquidityAddress = _liquidityAddress;
        developmentAddress = _developmentAddress;
    }
```

Remediation

Consider implementing **require** statements where appropriate to validate all user-controlled input, including constructor, to avoid the potential for erroneous values to result in unexpected behaviors or wasted gas.

Status: Acknowledged by the auditee.

Informational

8. Missing Events for Significant Transactions

Line	Code

211	function excludeFromFee(address account) public onlyOwner {
	_isExcludedFromFee[account] = true;
	}

215	function includeInFee(address account) public onlyOwner {
	_isExcludedFromFee[account] = false;
	}

219	function setTaxFeePercent(uint256 taxFee) external onlyOwner() {
	_taxFee = taxFee;
	}

223	function setLiquidityFeePercent(uint256 liquidityFee) external onlyOwner() {
	_liquidityFee = liquidityFee;
	}

227	function setDevelopmentFeePercent(uint256 developmentFee) external onlyOwner() {
	_developmentFee = developmentFee;
	}

231	function setMaxTxPercent(uint256 maxTxPercent) external onlyOwner() {
	_maxTxAmount = _tTotal.mul(maxTxPercent).div(
	10**2
	);
	}

Description

The missing event makes it difficult to track off-chain liquidity fee changes. An event should be emitted for significant transactions calling the following functions:

- excludeFromFee
- includeInFee
- setTaxFeePercent
- setLiquidityFeePercent
- setDevelopmentFeePercent
- setMaxTxPercent

Remediation
We recommend emitting an event to log the update of the following variables:

- _isExcludedFromFee
- _taxFee
- _liquidityFee
- _developmentFee
- _maxTxAmount

Status: Acknowledged by the auditee.

9. Order of Functions

Description
The following functions and function types should be re-ordered:
- The fallback function **receive() external payable {}.**
- The **view** and **pure** functions

Ordering helps readers identify which functions they can call and to find the constructor and fallback definitions easier.

Remediation
Functions should be grouped according to their visibility and ordered:

- constructor
- fallback function (if exists)
- external
- public
- internal
- private

Within a grouping, place the view and pure functions last.

Status: Acknowledged by the auditee.

10. Incorrect versions of Solidity & Different pragma directives are used

Pragma version>=0.6.0<0.8.0
Pragma version>=0.6.2<0.8.0
Pragma version 0.7.0

Description

solc frequently releases new compiler versions. Using an old version prevents access to new Solidity security checks. We also recommend avoiding complex pragma statements.

Remediation

Use one Solidity version and deploy with the following Solidity versions: **0.7.5** or **0.7.6**

Use a simple pragma version that allows any of these versions. Deploy with any of the following Solidity versions:

0.5.16 - 0.5.17
0.6.11 - 0.6.12
0.7.5 - 0.7.6

Status: Open

11. Misleading comments

Line	Code
383	restoreAllFee(); // this is very storage inefficien

Description

Wrong comments were found for the above code. A misunderstanding comment could influence code readability.

Remediation

We recommend correcting the comments or removing them.

Status: Fixed
The comment has been removed.

12. Declare variables and assign it to arrays

Description

Array of numbers **tInfo[4]** and **rInfor[4]** is being utilized in the entire project; this could influence code readability, in some cases, which may lead to bugs in the future.

Remediation

Declares a variable and assigns it to an array with the given initial values. Your code will be more readable if you give your variables easy-to-understand variable names.

Status: Acknowledged by the auditee.

13. Comparison to boolean constants

Line	Code
60	require(initialized == false, "LOLZImp: contract has already been initialized.");

Description

Boolean constants of the variable **initialized** can be used directly and do not need to be compared to true or false.

Remediation

We recommend changing the comparison to **require(!initialized, "LOLZImp: contract has already been initialized.");**

Status: Fixed

Comparison to boolean has been removed.

14. State Variable Default Visibility

Line	Code
60	bool inSwapAndLiquify;

Description
The Visibility of the **inSwapAndLiquify** variable is not defined.
Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.
The default is internal for state variables, but it should be made explicit.

Remediation
We recommend adding the visibility for the state variable of **inSwapAndLiquify.**
Variables can be specified as being **public, internal** or **private**.
Explicitly define visibility for all state variables.

Status: Fixed
The variable has been removed as described in section 4

15. Conformance to Solidity naming conventions

Line	Code
23	uint256 private constant _tTotal = 1000000000 * 10**6 * 10**9;

Description
Constants should be named with all capital letters with underscores separating words. Examples: MAX_BLOCKS, TOKEN_NAME, TOKEN_TICKER, CONTRACT_VERSION

Remediation
Follow the Solidity naming convention.

Status: Acknowledged by the auditee.

16. Public function that could be declared external

Description

The following **public** functions that are never called by the contract should be declared **external** to save gas:

- initialize()
- deliver()
- reflectionFromToken()
- excludeFromReward()
- includeInReward()
- isExcludedFromReward()
- excludeFromFee()
- includeInFee()

Remediation

Use the external attribute for functions never called from the contract.

Status: Acknowledged by the auditee.

17. Not using delete to zero values

Description

In the **removeAllFee** function, the following variables are manually replaced with Zero:

- _taxFee
- _liquidityFee
- _developmentFee

Remediation

To simplify the code and clarify intent, consider using **delete** instead.

Status: Acknowledged by the auditee.

18. Conformance to Solidity naming conventions

Description

It is extremely difficult to locate any contracts or functions, as they lack documentation. One consequence of this is that reviewers' understanding of the code's intention is impeded, which is significant because it is necessary to accurately determine both security and correctness.

They are additionally more readable and easier to maintain when wrapped in docstrings. The functions should be documented so that users can understand the purpose or intention of each function, as well as the situations in which it may fail, who is allowed to call it, what values it returns, and what events it emits.

Remediation

Consider thoroughly documenting all functions (and their parameters) that are part of the contracts' public API. Functions implementing sensitive functionality, even if those are not public, should be clearly documented as well. When writing docstrings, consider following the Ethereum Natural Specification Format (NatSpec).

Status: Acknowledged by the auditee.

Functional test

Function Names	Testing results
approve	Passed
decreaseAllowance	Passed
deliver	Passed
excludeFromFee	Passed
excludeFromReward	Passed
includeInFee	Passed
includeInReward	Passed
increaseAllowance	Passed
Initialize	Passed
renounceOwnership	Passed
setDevelopmentFeePercent	Passed
setLiquidityFeePercent	Passed
setMaxTxPercent	Passed
transfer	Passed
transferFrom	Passed
transferOwnership	Passed
name	Passed
symbol	Passed
totalSupply	Passed
reflectionFromToken	Passed

Function Names	Testing results
isExcludedFromReward	Passed
excludeFromReward	Passed
includeInReward	Passed
setTaxFeePercent	Passed
balanceOf	Passed
isExcludedFromFee	Passed

Automated Testing

Slither

```
INFO:Detectors:
LOLZImp.reflectionFromToken(uint256,bool).rInfo_scope_0 (LOLZImp.sol#163) is a storage variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-storage-variables
INFO:Detectors:
Contract locking ether found:
        Contract LOLZImp (LOLZImp.sol#9-430) has payable functions:
         - LOLZImp.receive() (LOLZImp.sol#237)
        But does not have a function to withdraw the ether
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#contracts-that-lock-ether
INFO:Detectors:
LOLZImp.allowance(address,address).owner (LOLZImp.sol#114) shadows:
        - Ownable.owner() (Ownable.sol#35-37) (function)
LOLZImp._approve(address,address,uint256).owner (LOLZImp.sol#338) shadows:
        - Ownable.owner() (Ownable.sol#35-37) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
LOLZImp.constructor(address,address)._liquidityAddress (LOLZImp.sol#54) lacks a zero-check on :
                - liquidityAddress = _liquidityAddress (LOLZImp.sol#55)
LOLZImp.constructor(address,address)._developmentAddress (LOLZImp.sol#54) lacks a zero-check on :
                - developmentAddress = _developmentAddress (LOLZImp.sol#56)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Variable 'LOLZImp.reflectionFromToken(uint256,bool).rInfo (LOLZImp.sol#160)' in LOLZImp.reflectionFromToken(uint256,bool) (LOLZImp.sol#157-166) potentially used before declaration: (rInfo) = _
getValues(tAmount) (LOLZImp.sol#163)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
INFO:Detectors:
Address.isContract(address) (openzeppelin/contracts/utils/Address.sol#26-35) uses assembly
        - INLINE ASM (openzeppelin/contracts/utils/Address.sol#33)
Address._verifyCallResult(bool,bytes,string) (openzeppelin/contracts/utils/Address.sol#171-188) uses assembly
        - INLINE ASM (openzeppelin/contracts/utils/Address.sol#180-183)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
LOLZImp.initialize() (LOLZImp.sol#59-86) compares to a boolean constant:
        -require(bool,string)(initialized == false,LOLZImp: contract has already been initialized.) (LOLZImp.sol#60)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Different versions of Solidity is used:
        - Version used: ['0.7.0', '>=0.6.0<0.8.0', '>=0.6.2<0.8.0']
        - 0.7.0 (LOLZImp.sol#1)
        - >=0.6.0<0.8.0 (Ownable.sol#3)
        - >=0.6.0<0.8.0 (openzeppelin/contracts/math/SafeMath.sol#3)
        - >=0.6.0<0.8.0 (openzeppelin/contracts/token/ERC20/IERC20.sol#3)
        - >=0.6.2<0.8.0 (openzeppelin/contracts/utils/Address.sol#3)
        - >=0.6.0<0.8.0 (openzeppelin/contracts/utils/Context.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
increaseAllowance(address,uint256) should be declared external:
        - LOLZImp.increaseAllowance(address,uint256) (LOLZImp.sol#129-132)
decreaseAllowance(address,uint256) should be declared external:
        - LOLZImp.decreaseAllowance(address,uint256) (LOLZImp.sol#134-137)
isExcludedFromReward(address) should be declared external:
        - LOLZImp.isExcludedFromReward(address) (LOLZImp.sol#139-141)
totalFees() should be declared external:
        - LOLZImp.totalFees() (LOLZImp.sol#143-145)
deliver(uint256) should be declared external:
        - LOLZImp.deliver(uint256) (LOLZImp.sol#147-155)
reflectionFromToken(uint256,bool) should be declared external:
        - LOLZImp.reflectionFromToken(uint256,bool) (LOLZImp.sol#157-166)
excludeFromReward(address) should be declared external:
        - LOLZImp.excludeFromReward(address) (LOLZImp.sol#174-181)
excludeFromFee(address) should be declared external:
        - LOLZImp.excludeFromFee(address) (LOLZImp.sol#211-213)
includeInFee(address) should be declared external:
        - LOLZImp.includeInFee(address) (LOLZImp.sol#215-217)
isExcludedFromFee(address) should be declared external:
        - LOLZImp.isExcludedFromFee(address) (LOLZImp.sol#334-336)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (Ownable.sol#54-57)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (Ownable.sol#63-67)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

```
INFO:Detectors:
Redundant expression "this (openzeppelin/contracts/utils/Context.sol#21)" inContext (openzeppelin/contracts/utils/Context.sol#15-24)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Variable LOLZImp._directToTeam(uint256,uint256).rDevelopment (LOLZImp.sol#245) is too similar to LOLZImp._directToTeam(uint256,uint256).tDevelopment (LOLZImp.sol#24
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
LOLZImp.initialize() (LOLZImp.sol#59-86) uses literals with too many digits:
        - _maxTxAmount = 5000000 * 10 ** 50 * 10 ** 9 (LOLZImp.sol#81)
LOLZImp.slitherConstructorConstantVariables() (LOLZImp.sol#9-430) uses literals with too many digits:
        - _tTotal = 1000000000 * 10 ** 6 * 10 ** 9 (LOLZImp.sol#23)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
LOLZImp.inSwapAndLiquify (LOLZImp.sol#43) is never used in LOLZImp (LOLZImp.sol#9-430)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
LOLZImp.inSwapAndLiquify (LOLZImp.sol#43) should be constant
LOLZImp.swapAndLiquifyEnabled (LOLZImp.sol#44) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
initialize() should be declared external:
        - LOLZImp.initialize() (LOLZImp.sol#59-86)
name() should be declared external:
        - LOLZImp.name() (LOLZImp.sol#88-90)
symbol() should be declared external:
        - LOLZImp.symbol() (LOLZImp.sol#92-94)
decimals() should be declared external:
        - LOLZImp.decimals() (LOLZImp.sol#96-98)
totalSupply() should be declared external:
        - LOLZImp.totalSupply() (LOLZImp.sol#100-102)
balanceOf(address) should be declared external:
        - LOLZImp.balanceOf(address) (LOLZImp.sol#104-107)
transfer(address,uint256) should be declared external:
        - LOLZImp.transfer(address,uint256) (LOLZImp.sol#109-112)
allowance(address,address) should be declared external:
        - LOLZImp.allowance(address,address) (LOLZImp.sol#114-116)
approve(address,uint256) should be declared external:
        - LOLZImp.approve(address,uint256) (LOLZImp.sol#118-121)
transferFrom(address,address,uint256) should be declared external:
        - LOLZImp.transferFrom(address,address,uint256) (LOLZImp.sol#123-127)
increaseAllowance(address,uint256) should be declared external:
        - LOLZImp.increaseAllowance(address,uint256) (LOLZImp.sol#129-132)
decreaseAllowance(address,uint256) should be declared external:
        - LOLZImp.decreaseAllowance(address,uint256) (LOLZImp.sol#134-137)
isExcludedFromReward(address) should be declared external:
        - LOLZImp.isExcludedFromReward(address) (LOLZImp.sol#139-141)
totalFees() should be declared external:
        - LOLZImp.totalFees() (LOLZImp.sol#143-145)
INFO:Detectors:
Address._verifyCallResult(bool,bytes,string) (openzeppelin/contracts/utils/Address.sol#171-188) is never used and should be removed
Address.functionCall(address,bytes) (openzeppelin/contracts/utils/Address.sol#79-81) is never used and should be removed
Address.functionCall(address,bytes,string) (openzeppelin/contracts/utils/Address.sol#89-91) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (openzeppelin/contracts/utils/Address.sol#104-106) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (openzeppelin/contracts/utils/Address.sol#114-121) is never used and should be removed
Address.functionDelegateCall(address,bytes) (openzeppelin/contracts/utils/Address.sol#153-155) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (openzeppelin/contracts/utils/Address.sol#163-169) is never used and should be removed
Address.functionStaticCall(address,bytes) (openzeppelin/contracts/utils/Address.sol#129-131) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (openzeppelin/contracts/utils/Address.sol#139-145) is never used and should be removed
Address.isContract(address) (openzeppelin/contracts/utils/Address.sol#26-35) is never used and should be removed
Address.sendValue(address,uint256) (openzeppelin/contracts/utils/Address.sol#53-59) is never used and should be removed
Context._msgData() (openzeppelin/contracts/utils/Context.sol#20-23) is never used and should be removed
SafeMath.div(uint256,uint256,string) (openzeppelin/contracts/math/SafeMath.sol#190-193) is never used and should be removed
SafeMath.mod(uint256,uint256) (openzeppelin/contracts/math/SafeMath.sol#152-155) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (openzeppelin/contracts/math/SafeMath.sol#210-213) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (openzeppelin/contracts/math/SafeMath.sol#24-28) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (openzeppelin/contracts/math/SafeMath.sol#60-63) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (openzeppelin/contracts/math/SafeMath.sol#70-73) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (openzeppelin/contracts/math/SafeMath.sol#45-53) is never used and should be removed
SafeMath.trySub(uint256,uint256) (openzeppelin/contracts/math/SafeMath.sol#35-38) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version0.7.0 (LOLZImp.sol#1) allows old versions
Pragma version>=0.6.0<0.8.0 (Ownable.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (openzeppelin/contracts/math/SafeMath.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (openzeppelin/contracts/token/ERC20/IERC20.sol#3) is too complex
Pragma version>=0.6.2<0.8.0 (openzeppelin/contracts/utils/Address.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (openzeppelin/contracts/utils/Context.sol#3) is too complex
solc-0.7.0 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) (openzeppelin/contracts/utils/Address.sol#53-59):
        - (success) = recipient.call{value: amount}() (openzeppelin/contracts/utils/Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (openzeppelin/contracts/utils/Address.sol#114-121):
        - (success,returndata) = target.call{value: value}(data) (openzeppelin/contracts/utils/Address.sol#119)
Low level call in Address.functionStaticCall(address,bytes,string) (openzeppelin/contracts/utils/Address.sol#139-145):
        - (success,returndata) = target.staticcall(data) (openzeppelin/contracts/utils/Address.sol#143)
Low level call in Address.functionDelegateCall(address,bytes,string) (openzeppelin/contracts/utils/Address.sol#163-169):
        - (success,returndata) = target.delegatecall(data) (openzeppelin/contracts/utils/Address.sol#167)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter LOLZImp.calculateTaxFee(uint256)._amount (LOLZImp.sol#297) is not in mixedCase
Parameter LOLZImp.calculateLiquidityFee(uint256)._amount (LOLZImp.sol#303) is not in mixedCase
Parameter LOLZImp.calculateDevelopmentFee(uint256)._amount (LOLZImp.sol#310) is not in mixedCase
Variable LOLZImp._isExcludedFromFee (LOLZImp.sol#17) is not in mixedCase
Constant LOLZImp._tTotal (LOLZImp.sol#23) is not in UPPER_CASE_WITH_UNDERSCORES
Variable LOLZImp._taxFee (LOLZImp.sol#31) is not in mixedCase
```

Mythril

```
enderphan@enderphan contracts % myth a LOLZImp.sol
The analysis was completed successfully. No issues were detected.
```

THEO

```
enderphan@enderphan solidity-pentest % theo --rpc-http http://127.0.0.1:8545
The account's private key (input hidden)
>
Contract to interact with
> 0x65BabD37318c4c8567a53d4956F538deB4b2DBE0
Scanning for exploits in contract: 0x65BabD37318c4c8567a53d4956F538deB4b2DBE0
Connecting to HTTP: http://127.0.0.1:8545.
No exploits found. You're going to need to load some exploits.

Tools available in the console:
- `exploits` is an array of loaded exploits found by Mythril or read from a file
- `w3` an initialized instance of web3py for the provided HTTP RPC endpoint
- `dump()` writing a json representation of an object to a local file

Check the readme for more info:
https://github.com/cleanunicorn/theo

Theo version v0.8.2.
```

Solhint Linter

```
contracts/LOLZImp.sol:1:1: Error: Compiler version 0.7.5 does not satisfy
the r semver requirement

contracts/LOLZImp.sol:9:1: Error: Contract has 21 states declarations but
allowed no more than 15

contracts/LOLZImp.sol:23:30: Error: Constant name must be in capitalized
SNAKE_CASE

contracts/LOLZImp.sol:43:5: Error: Explicitly mark visibility of state

contracts/LOLZImp.sol:237:32: Error: Code contains empty blocks
```

Solidity Static Analysis

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Address.functionCallWithValue(address,bytes,uint256,string): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 114:4:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases.
Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 33:8:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases.
Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 180:16:

Low level calls:

Use of "call": should be avoided whenever possible.
It can lead to unexpected behavior if return value is not handled properly.
Please use Direct Calls via specifying the called contract's interface.
more
Pos: 57:27:

Low level calls:

Use of "call": should be avoided whenever possible.
It can lead to unexpected behavior if return value is not handled properly.
Please use Direct Calls via specifying the called contract's interface.
more
Pos: 119:50:

Low level calls:

Use of "delegatecall": should be avoided whenever possible.
External code, that is called can change the state of the calling contract and send ether from the caller's balance.
If this is wanted behaviour, use the Solidity library feature if possible.
more
Pos: 167:50:

Disclaimer

Quillhash audit is not a security warranty, investment advice, or an endorsement of the LOLZTOKEN 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 LOLZTOKEN Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

Closing Summary

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

The audit showed several medium, low and informational severity issues. In the end, the majority of the issues were fixed and acknowledged by the Auditee. Other issues still remained unfixed as the internal team needs further discussion.

LoLz FINANCE

QuillAudits

Canada, India, Singapore and United Kingdom

audits.quillhash.com

audits@quillhash.com