Guard

Smart contracts security assessment

Final report

Tariff: Standard

Liquid AR

June 2022

0xguard.com

hello@0xguard.com

Contents

Introduction

The report has been prepared for Liquid Capital.

The 'Liquid AR' token is a rebase-like token with a fee on transfers. Part of the fees is directed to increase liquidity in the pairs.

The rebasing mechanism allows you to increase the balance of users. But at the same time, with an increase in the balance of users, the balance of tokens on the pair will increase, which will lead to a corresponding drop in the token rate.

The code is available at the @grapefi/token GitHub repository and was audited after the commit 8a8aaad.

Report Update

The contracts code was updated by the Liquid Capital team (commit af526d38) according to this report.

Name	Liquid AR
Audit date	2022-06-16 - 2022-06-20
Language	Solidity
Platform	Binance Smart Chain

Contracts checked

Name	Address
SafeMathInt	
Context	
Auth	
SafeMath	

ERC20Detailed

LAR

Procedure

We perform our audit according to the following procedure:

Automated analysis

- Scanning the project's smart contracts with several publicly available automated Solidity analysis tools
- Manual verification (reject or confirm) all the issues found by the tools

Manual audit

- Manually analyze smart contracts for security vulnerabilities
- Smart contracts' logic check

Known vulnerabilities checked

Title	Check result
Unencrypted Private Data On-Chain	passed
Code With No Effects	passed
Message call with hardcoded gas amount	passed
Typographical Error	passed
DoS With Block Gas Limit	passed
Presence of unused variables	passed
Incorrect Inheritance Order	passed
Requirement Violation	passed

Weak Sources of Randomness from Chain Attributes	passed
Shadowing State Variables	passed
Incorrect Constructor Name	passed
Block values as a proxy for time	passed
Authorization through tx.origin	passed
DoS with Failed Call	passed
Delegatecall to Untrusted Callee	passed
Use of Deprecated Solidity Functions	passed
Assert Violation	passed
State Variable Default Visibility	passed
Reentrancy	passed
Unprotected SELFDESTRUCT Instruction	passed
Unprotected Ether Withdrawal	passed
Unchecked Call Return Value	passed
Floating Pragma	passed
Outdated Compiler Version	passed
Integer Overflow and Underflow	passed
Function Default Visibility	passed

Classification of issue severity

High severity High severity issues can cause a significant or full loss of funds, change of contract ownership, major interference with contract logic. Such issues require immediate attention.

Medium severity	Medium severity issues do not pose an immediate risk, but can be detrimental to the client's reputation if exploited. Medium severity issues may lead to a contract failure and can be fixed by modifying the contract state or redeployment. Such issues require attention.
Low severity	Low severity issues do not cause significant destruction to the contract's functionality. Such issues are recommended to be taken into consideration.

Issues

High severity issues

1. Blocking transfers by the owner (LAR)

The contract owner can block all users' transfers by using the `setInitialDistributionFinished()` function. Thus, users will not be able to buy, sell or exchange their tokens.

Recommendation: It is necessary to restrict the owner's rights to call the function `setInitialDistributionFinished()` more than once.

2. Blacklist (LAR)

The contract owner has the ability to restrict transfers of any user at any time by using the `updateBlacklist()` function. Thus, users will not be able to buy, sell or exchange their tokens.

Recommendation: It is necessary to restrict the owner's rights to blacklist users.

Medium severity issues

No issues were found

Low severity issues

1. Unused functionality - FIXED (SafeMathInt)

The functions `abs()`, `max()`, `min()` are never used and can be removed to save gas on deployment.

Moreover, the library is used for integers (`int`), but the `max()`, `min()` functions are available for unsigned integers (`unit`) only.

2. Unused functionality - FIXED (Context)

1. The functions `_msgSender()`, `_msgData()` are never used and can be removed to save gas on deployment.

2. Using of the `this` statement in the `_msgData()` function is redundant.

Recommendation: Remove the unused contract to save gas on deployment.

3. State variable default visibility - FIXED (Auth)

The state variable `owner` L64 has default visibility. Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.

4. Gas optimization - FIXED (Auth)

Visibility of the functions `authorize()`, `unauthorize()`, `transferOwnership()` can be declared as '`external`' to save gas.

5. Unused functionality - FIXED (SafeMath)

The functions `mod()`, `max()`, `min()` are never used and can be removed to save gas on

deployment.

6. Gas optimization - FIXED (ERC20Detailed)

Visibility of the functions `name()`, `symbol()`, `decimals()` can be declared as '`external`' to save gas.

7. Incorrect naming - FIXED (LAR)

The state variable `usdtToken` on L341 stores the address of the BUSD token. Moreover, the local variable `pairBusd` on L380 has proper naming.

Recommendation: Consider replacing the variable name `usdtToken` with `busdToken`.

8. Incomplete functionality - FIXED (LAR)

The state variables `rebaseIndex`, `rebaseEpoch` are updated in the `updateRebaseIndex()` function. But their values are never used again.

9. Floating pragma - FIXED (LAR)

The general recommendation is that pragma should be fixed to the version you intend to deploy your contracts with. This helps to avoid deploying using an outdated compiler version and shields from possible bugs in future solidity releases.

10. Gas optimization - FIXED (LAR)

1. The variables `oneEEighteen`, `usdtToken` can be declared as '`constant`' to save gas.

2. Visibility of the functions `balanceOf()`, `markerPairAddress()`, `currentIndex()`, `getGonBalances()`, `getCirculatingSupply()`, `getCurrentTimestamp()` can be declared as 'external' to save gas.

3. The calculation of the value of the variable `gonSwapThreshold` on L369 can be simplified to the form: `TOTAL_GONS / 1000`.

4. The value of the `_markerPairs.length` on L844 can be stored in the local variable to save gas.

Conclusion

Liquid AR SafeMathInt, Context, Auth, SafeMath, ERC20Detailed, LAR contracts were audited. 2 high, 10 low severity issues were found.

According to this report 10 low issues were fixed by the Liquid Capital team.

We strongly recommend writing unit tests to have extensive coverage of the codebase minimize the possibility of bugs and ensure that everything works as expected.

The contracts are dependent on the owner's account. Users interacting with the contracts have to trust the owner.

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 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 without 0xGuard prior written consent.

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 0xGuard 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 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.

Slither output

```
LAR._addLiquidity(uint256,uint256) (contracts/LAR.sol#605-614) sends eth to arbitrary
user
        Dangerous calls:
        - router.addLiquidityETH{value: BNBAmount}
(address(this),tokenAmount,0,0,liquidityReceiver,block.timestamp) (contracts/
LAR.sol#606-613)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-
send-ether-to-arbitrary-destinations

Reentrancy in LAR._transferFrom(address,address,uint256) (contracts/LAR.sol#514-558):
        External calls:
        - swapBack() (contracts/LAR.sol#539)
                - router.addLiquidityETH{value: BNBAmount}
(address(this),tokenAmount,0,0,liquidityReceiver,block.timestamp) (contracts/
LAR.sol#606-613)
                - router.addLiquidity(address(this),usdtToken,tokenAmount,busdAmount,0,0
,liquidityReceiver,block.timestamp) (contracts/LAR.sol#619-628)
                - router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmou
nt,0,path,receiver,block.timestamp) (contracts/LAR.sol#652-658)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,
0,path,receiver,block.timestamp) (contracts/LAR.sol#638-644)
        External calls sending eth:
        - swapBack() (contracts/LAR.sol#539)
                - router.addLiquidityETH{value: BNBAmount}
(address(this),tokenAmount,0,0,liquidityReceiver,block.timestamp) (contracts/
LAR.sol#606-613)
        State variables written after the call(s):
        - _gonBalances[sender] = _gonBalances[sender].sub(gonAmount) (contracts/
LAR.sol#542)
        - _gonBalances[recipient] = _gonBalances[recipient].add(gonAmountReceived)
(contracts/LAR.sol#547-549)
        - gonAmountReceived = takeFee(sender,recipient,gonAmount) (contracts/
LAR.sol#544-546)
                - _gonBalances[address(this)] =
_gonBalances[address(this)].add(feeAmount) (contracts/LAR.sol#717-719)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities
```

```
LAR.swapBack() (contracts/LAR.sol#661-698) performs a multiplication on the result of a
division:
        -contractTokenBalance = _gonBalances[address(this)].div(_gonsPerFragment)
(contracts/LAR.sol#664-666)
        -amountToLiquify = contractTokenBalance.mul(liquidityFee.mul(2).add(sellFeeLiqui
dityAdded)).div(realTotalFee) (contracts/LAR.sol#668-670)
LAR.swapBack() (contracts/LAR.sol#661-698) performs a multiplication on the result of a
division:
        -contractTokenBalance = _gonBalances[address(this)].div(_gonsPerFragment)
(contracts/LAR.sol#664-666)
        -amountToRFV =
contractTokenBalance.mul(buyFeeRFV.mul(2).add(sellFeeRFVAdded)).div(realTotalFee)
(contracts/LAR.sol#672-674)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-
multiply

LAR.constructor() (contracts/LAR.sol#374-407) ignores return value by
IERC20(usdtToken).approve(address(router),type()(uint256).max) (contracts/LAR.sol#402)
LAR.constructor() (contracts/LAR.sol#374-407) ignores return value by
IERC20(usdtToken).approve(address(pairBusd),type()(uint256).max) (contracts/
LAR.sol#403)
LAR.constructor() (contracts/LAR.sol#374-407) ignores return value by
IERC20(usdtToken).approve(address(this),type()(uint256).max) (contracts/LAR.sol#404)
LAR._addLiquidity(uint256,uint256) (contracts/LAR.sol#605-614) ignores return value by
router.addLiquidityETH{value: BNBAmount}
(address(this),tokenAmount,0,0,liquidityReceiver,block.timestamp) (contracts/
LAR.sol#606-613)
LAR._addLiquidityBusd(uint256,uint256) (contracts/LAR.sol#616-629) ignores return value
by router.addLiquidity(address(this),usdtToken,tokenAmount,busdAmount,0,0,liquidityRecei
ver,block.timestamp) (contracts/LAR.sol#619-628)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

Auth.transferOwnership(address).adr (contracts/LAR.sol#98) lacks a zero-check on :
                - owner = adr (contracts/LAR.sol#99)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-
address-validation

LAR.manualSync() (contracts/LAR.sol#484-488) has external calls inside a loop:
ILiquidityProvider(_markerPairs[i]).sync() (contracts/LAR.sol#486)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-
```

```
a-loop

Reentrancy in LAR.constructor() (contracts/LAR.sol#374-407):
        External calls:
        - pair = IDEXFactory(router.factory()).createPair(address(this),router.WETH())
(contracts/LAR.sol#376-379)
        - pairBusd = IDEXFactory(router.factory()).createPair(address(this),usdtToken)
(contracts/LAR.sol#380-383)
        State variables written after the call(s):
        - _allowedFragments[address(this)][address(router)] = uint256(- 1) (contracts/
LAR.sol#385)
        - _allowedFragments[address(this)][pair] = uint256(- 1) (contracts/LAR.sol#386)
        - _allowedFragments[address(this)][address(this)] = uint256(- 1) (contracts/
LAR.sol#387)
        - _allowedFragments[address(this)][pairBusd] = type()(uint256).max (contracts/
LAR.sol#388)
        - _gonBalances[msg.sender] = TOTAL_GONS (contracts/LAR.sol#394)
        - _gonsPerFragment = TOTAL_GONS.div(_totalSupply) (contracts/LAR.sol#395)
        - _isFeeExempt[treasuryReceiver] = true (contracts/LAR.sol#397)
        - _isFeeExempt[riskFreeValueReceiver] = true (contracts/LAR.sol#398)
        - _isFeeExempt[address(this)] = true (contracts/LAR.sol#399)
        - _isFeeExempt[msg.sender] = true (contracts/LAR.sol#400)
        - setAutomatedMarketMakerPair(pair,true) (contracts/LAR.sol#390)
                - _markerPairCount ++ (contracts/LAR.sol#841)
        - setAutomatedMarketMakerPair(pairBusd,true) (contracts/LAR.sol#391)
                - _markerPairCount ++ (contracts/LAR.sol#841)
        - setAutomatedMarketMakerPair(pair,true) (contracts/LAR.sol#390)
                - _markerPairs.push(_pair) (contracts/LAR.sol#840)
                - _markerPairs[i] = _markerPairs[_markerPairs.length - 1] (contracts/
LAR.sol#846)
                - _markerPairs.pop() (contracts/LAR.sol#847)
        - setAutomatedMarketMakerPair(pairBusd,true) (contracts/LAR.sol#391)
                - _markerPairs.push(_pair) (contracts/LAR.sol#840)
                - _markerPairs[i] = _markerPairs[_markerPairs.length - 1] (contracts/
LAR.sol#846)
                - _markerPairs.pop() (contracts/LAR.sol#847)
        - _totalSupply = INITIAL_FRAGMENTS_SUPPLY (contracts/LAR.sol#393)
        - setAutomatedMarketMakerPair(pair,true) (contracts/LAR.sol#390)
                - automatedMarketMakerPairs[_pair] = _value (contracts/LAR.sol#837)
        - setAutomatedMarketMakerPair(pairBusd,true) (contracts/LAR.sol#391)
                - automatedMarketMakerPairs[_pair] = _value (contracts/LAR.sol#837)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-2

Reentrancy in LAR._swapAndLiquify(uint256) (contracts/LAR.sol#575-603):
        External calls:
        - _swapTokensForBNB(half,address(this)) (contracts/LAR.sol#582)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,
0,path,receiver,block.timestamp) (contracts/LAR.sol#638-644)
        - _addLiquidity(otherHalf,newBalance) (contracts/LAR.sol#586)
                - router.addLiquidityETH{value: BNBAmount}
(address(this),tokenAmount,0,0,liquidityReceiver,block.timestamp) (contracts/
LAR.sol#606-613)
        External calls sending eth:
        - _addLiquidity(otherHalf,newBalance) (contracts/LAR.sol#586)
                - router.addLiquidityETH{value: BNBAmount}
(address(this),tokenAmount,0,0,liquidityReceiver,block.timestamp) (contracts/
LAR.sol#606-613)
        Event emitted after the call(s):
        - SwapAndLiquify(half,newBalance,otherHalf) (contracts/LAR.sol#588)
Reentrancy in LAR._swapAndLiquify(uint256) (contracts/LAR.sol#575-603):
        External calls:
        - _swapTokensForBusd(half,address(this)) (contracts/LAR.sol#592)
                - router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmou
nt,0,path,receiver,block.timestamp) (contracts/LAR.sol#652-658)
        - _addLiquidityBusd(otherHalf,newBalance_scope_1) (contracts/LAR.sol#598)
                - router.addLiquidity(address(this),usdtToken,tokenAmount,busdAmount,0,0
,liquidityReceiver,block.timestamp) (contracts/LAR.sol#619-628)
        Event emitted after the call(s):
        - SwapAndLiquify(half,newBalance_scope_1,otherHalf) (contracts/LAR.sol#600)
Reentrancy in LAR._transferFrom(address,address,uint256) (contracts/LAR.sol#514-558):
        External calls:
        - swapBack() (contracts/LAR.sol#539)
                - router.addLiquidityETH{value: BNBAmount}
(address(this),tokenAmount,0,0,liquidityReceiver,block.timestamp) (contracts/
LAR.sol#606-613)
                - router.addLiquidity(address(this),usdtToken,tokenAmount,busdAmount,0,0
,liquidityReceiver,block.timestamp) (contracts/LAR.sol#619-628)
                - router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmou
nt,0,path,receiver,block.timestamp) (contracts/LAR.sol#652-658)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,
0,path,receiver,block.timestamp) (contracts/LAR.sol#638-644)
```

```
        External calls sending eth:
        - swapBack() (contracts/LAR.sol#539)
                - router.addLiquidityETH{value: BNBAmount}
(address(this),tokenAmount,0,0,liquidityReceiver,block.timestamp) (contracts/
LAR.sol#606-613)
        Event emitted after the call(s):
        - Transfer(sender,address(this),feeAmount.div(_gonsPerFragment)) (contracts/
LAR.sol#720)
                - gonAmountReceived = takeFee(sender,recipient,gonAmount) (contracts/
LAR.sol#544-546)
        - Transfer(sender,recipient,gonAmountReceived.div(_gonsPerFragment)) (contracts/
LAR.sol#551-555)
Reentrancy in LAR.constructor() (contracts/LAR.sol#374-407):
        External calls:
        - pair = IDEXFactory(router.factory()).createPair(address(this),router.WETH())
(contracts/LAR.sol#376-379)
        - pairBusd = IDEXFactory(router.factory()).createPair(address(this),usdtToken)
(contracts/LAR.sol#380-383)
        Event emitted after the call(s):
        - SetAutomatedMarketMakerPair(_pair,_value) (contracts/LAR.sol#853)
                - setAutomatedMarketMakerPair(pairBusd,true) (contracts/LAR.sol#391)
        - SetAutomatedMarketMakerPair(_pair,_value) (contracts/LAR.sol#853)
                - setAutomatedMarketMakerPair(pair,true) (contracts/LAR.sol#390)
Reentrancy in LAR.constructor() (contracts/LAR.sol#374-407):
        External calls:
        - pair = IDEXFactory(router.factory()).createPair(address(this),router.WETH())
(contracts/LAR.sol#376-379)
        - pairBusd = IDEXFactory(router.factory()).createPair(address(this),usdtToken)
(contracts/LAR.sol#380-383)
        - IERC20(usdtToken).approve(address(router),type()(uint256).max) (contracts/
LAR.sol#402)
        - IERC20(usdtToken).approve(address(pairBusd),type()(uint256).max) (contracts/
LAR.sol#403)
        - IERC20(usdtToken).approve(address(this),type()(uint256).max) (contracts/
LAR.sol#404)
        Event emitted after the call(s):
        - Transfer(address(0x0),msg.sender,_totalSupply) (contracts/LAR.sol#406)
Reentrancy in LAR.swapBack() (contracts/LAR.sol#661-698):
        External calls:
        - _swapAndLiquify(amountToLiquify) (contracts/LAR.sol#681)
                - router.addLiquidityETH{value: BNBAmount}
```

```
(address(this),tokenAmount,0,0,liquidityReceiver,block.timestamp) (contracts/
LAR.sol#606-613)
                - router.addLiquidity(address(this),usdtToken,tokenAmount,busdAmount,0,0
,liquidityReceiver,block.timestamp) (contracts/LAR.sol#619-628)
                - router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmou
nt,0,path,receiver,block.timestamp) (contracts/LAR.sol#652-658)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,
0,path,receiver,block.timestamp) (contracts/LAR.sol#638-644)
        - _swapTokensForBusd(amountToRFV,riskFreeValueReceiver) (contracts/LAR.sol#685)
                - router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmou
nt,0,path,receiver,block.timestamp) (contracts/LAR.sol#652-658)
        - _swapTokensForBusd(amountToTreasury,treasuryReceiver) (contracts/LAR.sol#689)
                - router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmou
nt,0,path,receiver,block.timestamp) (contracts/LAR.sol#652-658)
        External calls sending eth:
        - _swapAndLiquify(amountToLiquify) (contracts/LAR.sol#681)
                - router.addLiquidityETH{value: BNBAmount}
(address(this),tokenAmount,0,0,liquidityReceiver,block.timestamp) (contracts/
LAR.sol#606-613)
        Event emitted after the call(s):
        - SwapBack(contractTokenBalance,amountToLiquify,amountToRFV,amountToTreasury)
(contracts/LAR.sol#692-697)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-3

LAR.manualRebase() (contracts/LAR.sol#795-810) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(nextRebase <= block.timestamp,Not in time) (contracts/
LAR.sol#797)
        - nextRebase < block.timestamp && i < 100 (contracts/LAR.sol#807)
LAR.setNextRebase(uint256) (contracts/LAR.sol#987-994) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_nextRebase > block.timestamp,Next rebase can not be in
the past) (contracts/LAR.sol#988-991)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-
timestamp

LAR.swapping() (contracts/LAR.sol#354-359) compares to a boolean constant:
        -require(bool,string)(inSwap == false,In swap already) (contracts/LAR.sol#355)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-
equality
```

```
LAR.coreRebase(int256) (contracts/LAR.sol#771-793) has costly operations inside a loop:
        - _totalSupply = _totalSupply.sub(uint256(- supplyDelta)) (contracts/
LAR.sol#778)
LAR.coreRebase(int256) (contracts/LAR.sol#771-793) has costly operations inside a loop:
        - _totalSupply = MAX_SUPPLY (contracts/LAR.sol#784)
LAR.coreRebase(int256) (contracts/LAR.sol#771-793) has costly operations inside a loop:
        - _gonsPerFragment = TOTAL_GONS.div(_totalSupply) (contracts/LAR.sol#787)
LAR.updateRebaseIndex() (contracts/LAR.sol#812-825) has costly operations inside a
loop:
        - nextRebase += rebaseFrequency (contracts/LAR.sol#814)
LAR.updateRebaseIndex() (contracts/LAR.sol#812-825) has costly operations inside a
loop:
        - rebaseIndex = rebaseIndex.mul(oneEEighteen.add(oneEEighteen.mul(rewardYield).d
iv(REWARD_YIELD_DENOMINATOR))).div(oneEEighteen) (contracts/LAR.sol#816-822)
LAR.updateRebaseIndex() (contracts/LAR.sol#812-825) has costly operations inside a
loop:
        - rebaseEpoch += 1 (contracts/LAR.sol#824)
LAR.coreRebase(int256) (contracts/LAR.sol#771-793) has costly operations inside a loop:
        - _totalSupply = _totalSupply.add(uint256(supplyDelta)) (contracts/LAR.sol#780)
LAR.setAutomatedMarketMakerPair(address,bool) (contracts/LAR.sol#828-854) has costly
operations inside a loop:
        - _markerPairs.pop() (contracts/LAR.sol#847)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-
operations-inside-a-loop

Context._msgData() (contracts/LAR.sol#57-60) is never used and should be removed
Context._msgSender() (contracts/LAR.sol#53-55) is never used and should be removed
SafeMath.max(uint256,uint256) (contracts/LAR.sol#194-196) is never used and should be
removed
SafeMath.min(uint256,uint256) (contracts/LAR.sol#198-200) is never used and should be
removed
SafeMath.mod(uint256,uint256) (contracts/LAR.sol#188-192) is never used and should be
removed
SafeMathInt.abs(int256) (contracts/LAR.sol#37-41) is never used and should be removed
SafeMathInt.add(int256,int256) (contracts/LAR.sol#30-35) is never used and should be
removed
SafeMathInt.div(int256,int256) (contracts/LAR.sol#17-21) is never used and should be
removed
SafeMathInt.max(uint256,uint256) (contracts/LAR.sol#43-45) is never used and should be
removed
```

```
SafeMathInt.min(uint256,uint256) (contracts/LAR.sol#47-49) is never used and should be
removed
SafeMathInt.mul(int256,int256) (contracts/LAR.sol#9-15) is never used and should be
removed
SafeMathInt.sub(int256,int256) (contracts/LAR.sol#23-28) is never used and should be
removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

LAR.totalBuyFee (contracts/LAR.sol#349) is set pre-construction with a non-constant
function or state variable:
        - liquidityFee.add(treasuryFee).add(buyFeeRFV)
LAR.totalSellFee (contracts/LAR.sol#350) is set pre-construction with a non-constant
function or state variable:
        -
totalBuyFee.add(sellFeeTreasuryAdded).add(sellFeeRFVAdded).add(sellFeeLiquidityAdded)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-
initializing-state

Pragma version^0.7.4 (contracts/LAR.sol#3) allows old versions
solc-0.7.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-
versions-of-solidity

Function IDEXRouter.WETH() (contracts/LAR.sol#233) is not in mixedCase
Parameter LAR.checkFeeExempt(address)._addr (contracts/LAR.sol#436) is not in mixedCase
Parameter LAR.setAutomatedMarketMakerPair(address,bool)._pair (contracts/LAR.sol#828)
is not in mixedCase
Parameter LAR.setAutomatedMarketMakerPair(address,bool)._value (contracts/LAR.sol#828)
is not in mixedCase
Parameter LAR.setInitialDistributionFinished(bool)._value (contracts/LAR.sol#856) is
not in mixedCase
Parameter LAR.setFeeExempt(address,bool)._addr (contracts/LAR.sol#863) is not in
mixedCase
Parameter LAR.setFeeExempt(address,bool)._value (contracts/LAR.sol#863) is not in
mixedCase
Parameter LAR.setSwapBackSettings(bool,uint256,uint256)._enabled (contracts/
LAR.sol#871) is not in mixedCase
Parameter LAR.setSwapBackSettings(bool,uint256,uint256)._num (contracts/LAR.sol#872) is
not in mixedCase
Parameter LAR.setSwapBackSettings(bool,uint256,uint256)._denom (contracts/LAR.sol#873)
is not in mixedCase
```

```
Parameter LAR.setFeeReceivers(address,address,address)._liquidityReceiver (contracts/
LAR.sol#881) is not in mixedCase
Parameter LAR.setFeeReceivers(address,address,address)._treasuryReceiver (contracts/
LAR.sol#882) is not in mixedCase
Parameter LAR.setFeeReceivers(address,address,address)._riskFreeValueReceiver
(contracts/LAR.sol#883) is not in mixedCase
Parameter LAR.setFees(uint256,uint256,uint256,uint256,uint256,uint256)._liquidityFee
(contracts/LAR.sol#898) is not in mixedCase
Parameter LAR.setFees(uint256,uint256,uint256,uint256,uint256,uint256)._riskFreeValue
(contracts/LAR.sol#899) is not in mixedCase
Parameter LAR.setFees(uint256,uint256,uint256,uint256,uint256,uint256)._treasuryFee
(contracts/LAR.sol#900) is not in mixedCase
Parameter
LAR.setFees(uint256,uint256,uint256,uint256,uint256,uint256)._sellFeeTreasuryAdded
(contracts/LAR.sol#901) is not in mixedCase
Parameter LAR.setFees(uint256,uint256,uint256,uint256,uint256,uint256)._sellFeeRFVAdded
(contracts/LAR.sol#902) is not in mixedCase
Parameter
LAR.setFees(uint256,uint256,uint256,uint256,uint256,uint256)._sellFeeLiquidityAdded
(contracts/LAR.sol#903) is not in mixedCase
Parameter LAR.setRouterPair(address,address)._router (contracts/LAR.sol#936) is not in
mixedCase
Parameter LAR.setRouterPair(address,address)._pair (contracts/LAR.sol#936) is not in
mixedCase
Parameter LAR.setIsLiquidityInBnb(bool)._value (contracts/LAR.sol#949) is not in
mixedCase
Parameter LAR.clearStuckBalance(address)._receiver (contracts/LAR.sol#955) is not in
mixedCase
Parameter LAR.setRebaseFrequency(uint256)._rebaseFrequency (contracts/LAR.sol#972) is
not in mixedCase
Parameter LAR.setRewardYield(uint256,uint256)._rewardYield (contracts/LAR.sol#979) is
not in mixedCase
Parameter LAR.setRewardYield(uint256,uint256)._rewardYieldDenominator (contracts/
LAR.sol#980) is not in mixedCase
Parameter LAR.setNextRebase(uint256)._nextRebase (contracts/LAR.sol#987) is not in
mixedCase
Parameter LAR.setMaxSellTransaction(uint256)._maxTxn (contracts/LAR.sol#996) is not in
mixedCase
Parameter LAR.updateBlacklist(address,bool)._user (contracts/LAR.sol#1002) is not in
mixedCase
Parameter LAR.updateBlacklist(address,bool)._flag (contracts/LAR.sol#1002) is not in
mixedCase
```

```
Variable LAR.REWARD_YIELD_DENOMINATOR (contracts/LAR.sol#300) is not in mixedCase
Variable LAR._markerPairCount (contracts/LAR.sol#307) is not in mixedCase
Variable LAR._markerPairs (contracts/LAR.sol#309) is not in mixedCase
Variable LAR._isFeeExempt (contracts/LAR.sol#311) is not in mixedCase
Constant LAR.feeDenominator (contracts/LAR.sol#318) is not in
UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-
solidity-naming-conventions

Redundant expression "this (contracts/LAR.sol#58)" inContext (contracts/LAR.sol#52-61)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-
statements

Reentrancy in LAR.clearStuckBalance(address) (contracts/LAR.sol#955-961):
        External calls:
        - address(_receiver).transfer(balance) (contracts/LAR.sol#958)
        Event emitted after the call(s):
        - ClearStuckBalance(balance,_receiver,block.timestamp) (contracts/LAR.sol#959)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-4

Variable IDEXRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address
,uint256).amountADesired (contracts/LAR.sol#238) is too similar to IDEXRouter.addLiquidi
ty(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired
(contracts/LAR.sol#239)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-
are-too-similar

LAR.slitherConstructorVariables() (contracts/LAR.sol#290-1082) uses literals with too
many digits:
        - REWARD_YIELD_DENOMINATOR = 10000000000000000 (contracts/LAR.sol#300)
LAR.slitherConstructorVariables() (contracts/LAR.sol#290-1082) uses literals with too
many digits:
        - rewardYield = 4166700000000 (contracts/LAR.sol#301)
LAR.slitherConstructorConstantVariables() (contracts/LAR.sol#290-1082) uses literals
with too many digits:
        - INITIAL_FRAGMENTS_SUPPLY = 50000000 * 10 ** DECIMALS (contracts/
LAR.sol#322-323)
LAR.slitherConstructorConstantVariables() (contracts/LAR.sol#290-1082) uses literals
with too many digits:
        - DEAD = 0x000000000000000000000000000000000000dEaD (contracts/LAR.sol#328)
```

```
LAR.slitherConstructorConstantVariables() (contracts/LAR.sol#290-1082) uses literals
with too many digits:
        - ZERO = 0x0000000000000000000000000000000000000000 (contracts/LAR.sol#329)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-
digits

SafeMathInt.MAX_INT256 (contracts/LAR.sol#7) is never used in SafeMathInt (contracts/
LAR.sol#5-50)
LAR.factory (contracts/LAR.sol#339) is never used in LAR (contracts/LAR.sol#290-1082)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-
variable

LAR.oneEEighteen (contracts/LAR.sol#299) should be constant
LAR.usdtToken (contracts/LAR.sol#341) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-
variables-that-could-be-declared-constant

authorize(address) should be declared external:
        - Auth.authorize(address) (contracts/LAR.sol#80-83)
unauthorize(address) should be declared external:
        - Auth.unauthorize(address) (contracts/LAR.sol#85-88)
transferOwnership(address) should be declared external:
        - Auth.transferOwnership(address) (contracts/LAR.sol#98-103)
name() should be declared external:
        - ERC20Detailed.name() (contracts/LAR.sol#218-220)
symbol() should be declared external:
        - ERC20Detailed.symbol() (contracts/LAR.sol#222-224)
decimals() should be declared external:
        - ERC20Detailed.decimals() (contracts/LAR.sol#226-228)
balanceOf(address) should be declared external:
        - LAR.balanceOf(address) (contracts/LAR.sol#424-426)
markerPairAddress(uint256) should be declared external:
        - LAR.markerPairAddress(uint256) (contracts/LAR.sol#428-430)
currentIndex() should be declared external:
        - LAR.currentIndex() (contracts/LAR.sol#432-434)
getGonBalances() should be declared external:
        - LAR.getGonBalances() (contracts/LAR.sol#467-471)
getCirculatingSupply() should be declared external:
        - LAR.getCirculatingSupply() (contracts/LAR.sol#473-478)
getCurrentTimestamp() should be declared external:
        - LAR.getCurrentTimestamp() (contracts/LAR.sol#480-482)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-
function-that-could-be-declared-external
```

0x Guard