TechRate
June, 2022

REPORT ON A HACKATHON

Hackathon 22-23 June 2022 Details

Audited project

REX

Deployer address

Not deployed

Client contacts:

REX team

Blockchain

Binance Smart Chain

Project website:

https://rex.io

Table of contents

- Manual audit Section
  - Auditor 1 issues
  - Auditor 2 issues
- Codebase analyzing tools Section
  - Slither
  - Mythril
- Unit tests Section
  - Unit tests overview
- Contracts Summary
  - Issues checking Status
  - Summary
- Attachments
  - Slither output
  - Mythril output

Manual audit Section

Auditor 1 issues

Issue	Severity	Location	Description
Overprice fee	Informational	DEX	2% fee taking if user is not a MREX HOLDER
Rounding error	Informational	REX	In _stakesShares() calculation with division, it goes first. In Solidity we don't have floating points, but instead we get rounding errors.
Daily snapshot	Informational	REX	_dailySnapshotPoint(_currentRxDay()) could be invoked as a modifier of the necessary functions.
Mismatch array lengths	Low	AIRDROP TREX REX	initClaimables(), createStakeBatch() functions don't compare lengths of _address list and _amount list
Randomness issue	Low	RDA	The way random addresses are chosen and shuffle may leave most part of list addresses at theirs positions, due to predictable BigPayDay loop their chances to get the bonus are increased.

Auditor 2 issues

Issue	Severity	Location	Description
Error return in view function	Informational	REX	_checkRewardAmountbyID() returns error if there are several days without Snapshot and calculating day is between last snapshot and current days.
Lengths mismatch	Low	REX, AIRDROP, TREX	In functions createStakeBatch() , initClaimables() there is no checking if lengths are equal. This can lead to wrong recording of stakes.
“Transfer” using	Low	RDA	_generateSupplyAndCheckPools() uses “transfer” to send BNB. “Transfer” and “send” should be avoided to transfer BNB. Use “call” with reentrancy protection instead.

Codebase analyzing tools Section

1. Slither

Dex Contract:

Number of lines: 373 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 5 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 0
Number of informational issues: 25
Number of low issues: 3
Number of medium issues: 3
Number of high issues: 0

AIRDROP Contract:

Number of lines: 339 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 4 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 0
Number of informational issues: 23
Number of low issues: 2
Number of medium issues: 1
Number of high issues: 0

TREX Contract:

Number of lines: 1043 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 8 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 0
Number of informational issues: 30
Number of low issues: 9
Number of medium issues: 4
Number of high issues: 4

REX Contract:

Number of lines: 1664 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 19 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 0
Number of informational issues: 60
Number of low issues: 24
Number of medium issues: 20
Number of high issues: 0

RDA Contract:

Number of lines: 1607 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 7 (+ 0 in dependencies, + 0 tests)
Number of optimization issues: 0
Number of informational issues: 44
Number of low issues: 33
Number of medium issues: 18
Number of high issues: 18

See the attachments for more info.

1. Mythrill

Dex Contract:

Number of lines: 373 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 5 (+ 0 in dependencies, + 0 tests)

Number of low issues: 26
Number of medium issues: 4
Number of high issues: 0

AIRDROP Contract:

Number of lines: 339 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 4 (+ 0 in dependencies, + 0 tests)

Number of low issues: 0
Number of medium issues: 1
Number of high issues: 0

TREX Contract:

Number of lines: 1043 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 8 (+ 0 in dependencies, + 0 tests)

Number of low issues: 9
Number of medium issues: 1
Number of high issues: 0

REX Contract:

Number of lines: 1664 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 19 (+ 0 in dependencies, + 0 tests)

Number of low issues: 10
Number of medium issues: 0
Number of high issues: 0

RDA Contract:

Number of lines: 1607 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 7 (+ 0 in dependencies, + 0 tests)

Number of low issues: 10
Number of medium issues: 0
Number of high issues: 0

Input data:
Restriction of 10000 transactions and 1000 timeout.

See the attachments for more info.

Unit tests Section

Unit tests overview

Unit tests are typically automated tests written and run by software developers and auditors to ensure that a section of a smart contract meets its design and behaves as intended. We create tests first for the smallest testable units, then for the compound interaction between them, building up comprehensive tests for complex applications.

Contract	Function	Cases	Status
REX	initRexContracts	1	Passed
	CreateStake	5	Passed
	CreateStakeBatch	2	Passed
	endStake	3	Passed
	manualSnapshotOneDay	4	Passed
	_checkRewardAmountbyID	5	Passed partially
	withdrawRewards	5	Passed
	offerStake	2	Passed
	splitStake	1	Passed
	transferStake	2	Passed
	transfer (REX)	2	Passed
	BalanceOf	5	Passed
RDA	donateBUSD	2	Passed
	sendLiquidityBUSD	4	Passed
	withdrawLPTokens	7	Passed
	withdrawableLPTokens	7	Passed
	triggerDailyRoutineOneDay	6	Passed
	getBPDCount	4	Passed
	_createUserBPD	5	Passed
	claimStakeFromDonations	4	Passed
	claimRexFromDonations	3	Passed
	claimRexFromReferrals	4	Passed
	claimBusdFromReferrals	5	Passed
	claimBusdFromBPD	4	Passed
	claimBusdFromTREASURY	3	Passed
	myClaimableRexFromDonations	4	Passed
	myClaimableRexFromReferrals	2	Passed
	getActualUnhitByRandom	7	Passed
	auctionSupplyOnDay	5	Passed
	auctionStatsOfDay	1	Passed
	_currentRxDay	1	Passed
	withdrawTokensAfterContractEnd	1	Passed

AIRDROP	initContract	1	Passed
	initClaimables	2	Passed
	revokeAccess	1	Passed
	addressCanClaimNow	4	Passed
	addressCanStakeNow	4	Passed
	addressIsInAirdropList	3	Passed
	claimAirdropTokens	2	Passed
	claimAirdropStake	2	Passed
	_currentRxDay	1	Passed
DEX	initRexContract	1	Passed
	revokeAccess	1	Passed
	listStake	3	Passed
	buyStakeFromList	3	Passed
	revokeOffer	3	Passed
	_currentRxDay	1	Passed
	dataOfOfferNumber	1	Passed
TREX	transfer	3	Passed
	balanceOf	3	Passed
	LAUNCH_TIME	1	Passed
	initClaimables	1	Passed
	getHalving	2	Passed
	_daysFromStart	3	Passed
	_getNeededAirdropReserve	4	Passed
	buyOneTREX	6	Passed
	sellAirdroppedTREX	7	Passed
	noOfTREXSellable	3	Passed
	canClaimAirdropNow	2	Passed
	claimAirdrop	2	Passed

Contracts Summary

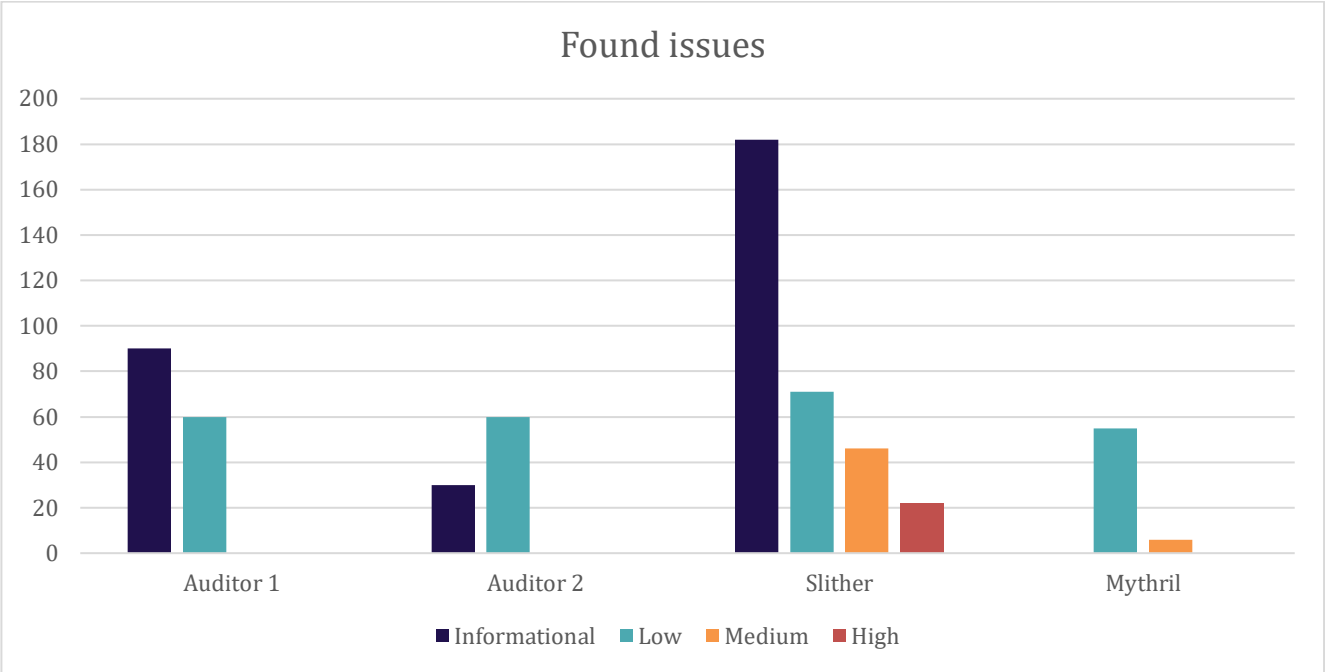
Issues Checking Status

	Issue description	Checking status
1.	**Compiler errors.** This type of issues refers to a state when a compiler fails due to errors or discrepancy in the code.	Passed
2.	**Race conditions. Cross-function race conditions.** One of the major dangers of calling external contracts is that they can take over the control flow, and make changes to your data that the calling function Is not expecting. Ex.: attack using two different functions that share the same state.	Passed
3.	**Possible delays in data delivery.** Delays in providing data that could affect contract logic.	Passed
4.	**Oracle calls.** Oracles retrieve and verify external data for blockchains and smart contracts through methods such as web APIs or market data feeds. The type of data required by smart contracts can include information on price feeds, weather information, or even random number generation for gambling. Leveraging oracles consists of querying the data source for specific information and subsequently connecting to that source to interface between the blockchain and the data feed. Marks that calls are not malicious and affect properly the contract logic.	Passed
5.	**Front running.** Since all transactions are visible in the mempool for a short while before being executed, observers of the network can see and react to an action before it is included in a block. Front-running, coined originally for traditional financial markets, is the race to order the chaos to the winner's benefit. In financial markets, the flow of information gave birth to intermediaries that could simply profit by being the first to know and react to some information. These attacks mostly had been within stock market deals and early domain registries.	Passed
6.	**Timestamp dependence.** Timestamp of the block can be manipulated by the	Passed

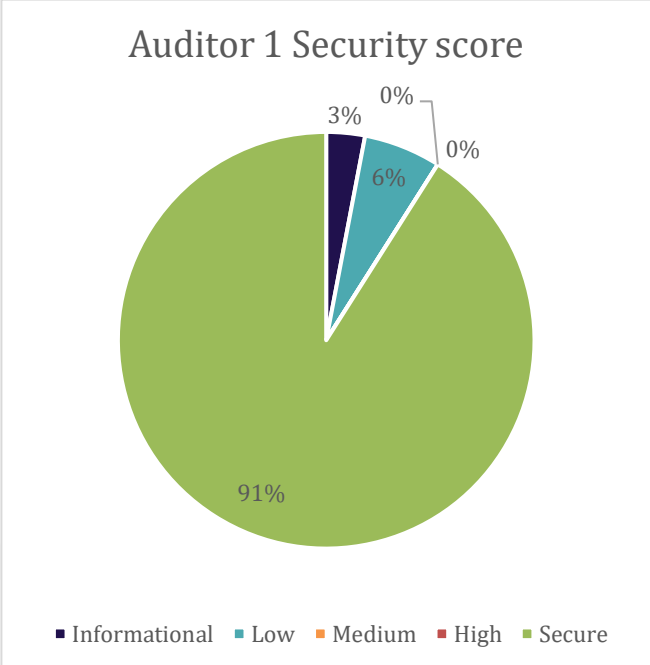
	miner, and all direct and indirect uses of the timestamp should be considered. Block numbers and average block time can be used to estimate time, but this is not future proof as block times may change.	
7.	**Integer Overflow and Underflow.** If a uint is made to be less than zero, it will cause an underflow and get set to its maximum value. The same is true for overflow. Be careful with the smaller data-types like uint8, uint16, uint24...etc: even more they can easily hit their maximum value.	Passed
8.	**DoS with Revert.** An attack, that causes a permanent denial of service to the contract rendering it useless.	Passed
9.	**DoS with block gas limit.** Each block has an upper bound on the amount of gas that can be spent, and thus the amount computation that can be done. This is the Block Gas Limit. If the gas spent exceeds this limit, the transaction will fail.	Passed
10.	**Methods execution permissions.** Modifiers and other restrictions/allowances to owner to abuse authority or to user to call functions they should not have access for.	Passed
11.	**Economy model of the contract.** Marks that the contract economy is not falling or balances, cap, pool values...etc not going out of range.	Informational
12.	**The impact of the exchange rate on the logic.** When currencies are not backed by other valued assets, part of their price volatility may arise from self-fulfilling expectations of a speculative attack. Mark shows the stability of the contract economics over the impact of the exchange rate.	Passed
13.	**Private user data leaks.** Marks the user data are safe and not accessible by any other address.	Passed
14.	**Malicious Event log.** Logging events that don't reflect the real actions.	Passed
15.	**Scoping and Declarations.** When variables and other items are declared not in the code block, they become visible before the declaration.	Passed

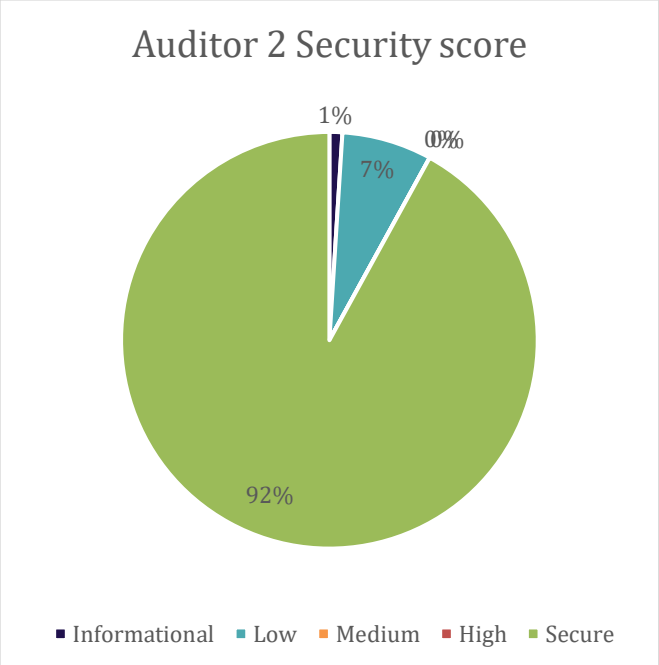
	Such declaration can occur in functions, user-defined types and contracts. Marks that variables declaration is proper and doesn't shadow the other.	
16.	**Uninitialized storage pointers.** The EVM stores data either as storage or as memory. Understanding exactly how this is done and the default types for local variables of functions is highly recommended when developing contracts. This is because it is possible to produce vulnerable contracts by inappropriately initializing variables.	Passed
17.	**Arithmetic accuracy.** Marks that contract's calculations are properly realized according to solidity version.	Low issues
18.	**Design Logic.** Contract's logical issues that affect their proper work.	Informational and Low issues
19.	**Reentrancy.** Functions that can be called repeatedly before the first function call completes could lead to destructive interaction between different function calls.	Passed
20.	**Safe Open Zeppelin contracts implementation and usage.** Mark shows the use of proven and trusted secure library mechanisms.	Passed
21.	**Fallback function security.** Since the fallback functions is not only called for plain ether transfers (without data) but also when no other function matches, you should check that the data is empty if the fallback function is intended to be used only for the purpose of logging received Ether. Otherwise, callers will not notice if your contract is used incorrectly and functions that do not exist are called.	Passed

Summary

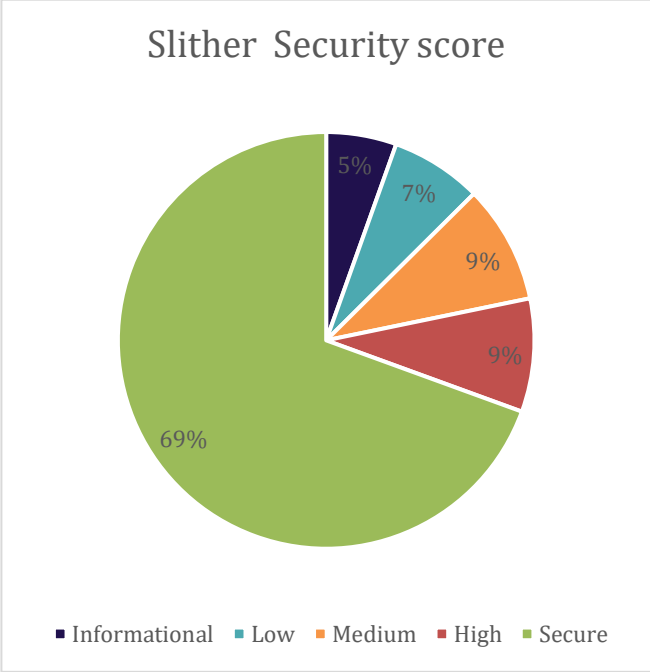


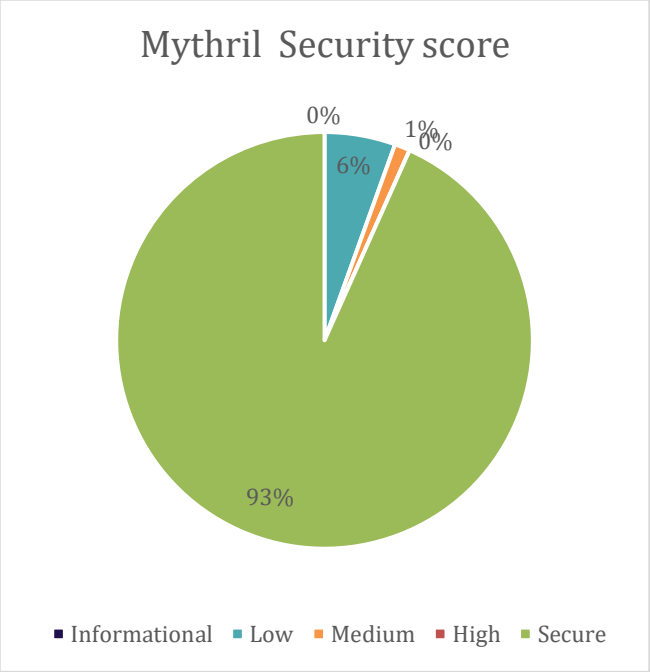
*using balancing coefficient





*using balancing coefficient





*using balancing coefficient

Audited 5 files: REX_TREX.sol, REX_DEX.sol, REX_AIRDROP.sol, REX_REX.sol, REX_RDA.sol (43 contracts)

During the hackathon the smart contracts were analyzed using complex approach to find all possible vulnerabilities. Both manual and machine based methodology was realized under the auditing and bug finding process. Auto test service results and machine based analysis results are listed in the report to get familiar with. Manual part consists of both: a line by line audit of the code and covering the functions with unit tests and testing the interaction of contracts among different scenarios. Found issues are presented above. According to the based complex analysis smart contracts contain low severity issues to put an eye on.

Attachments Section

1. Slither Output

Dex Contract:

Medium:

RexDEX.buyStakeFromList(uint256) (REX_dex.sol#208-242) uses a dangerous strict equality:

- MREX_TOKEN.balanceOf(msg.sender) == 0 (REX_dex.sol#225)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Contract locking ether found:

Contract RexDEX (REX_dex.sol#103-297) has payable functions:

- RexDEX.receive() (REX_dex.sol#152)
- RexDEX.fallback() (REX_dex.sol#153)

But does not have a function to withdraw the ether

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#contracts-that-lock-ether

RexDEX.listStake(address,uint32,uint32,uint256,bytes16)._offer (REX_dex.sol#186) is a local variable never initialized

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

Low & Informational:

Reentrancy in RexDEX.buyStakeFromList(uint256) (REX_dex.sol#208-242):

External calls:

- require(bool,string)(BUSD_TOKEN.transferFrom(msg.sender,MARKETING_ADDR,_fee),DEX: M-Fee transfer failed.) (REX_dex.sol#228)
- require(bool,string)(BUSD_TOKEN.transferFrom(msg.sender,DEVELOPMENT_ADDR,_fee),DEX: D-Fee transfer failed.) (REX_dex.sol#229)
- require(bool,string)(BUSD_TOKEN.transferFrom(msg.sender,_offer.staker,_offer.offerPrice),DEX: Transfer of BUSD failed.) (REX_dex.sol#233)

State variables written after the call(s):

- gl.totalFulfilledOffers = gl.totalFulfilledOffers.add(1) (REX_dex.sol#236)
- gl.totalTradingVolume = gl.totalTradingVolume.add(_offer.offerPrice) (REX_dex.sol#237)
- gl.totalActiveOffers = gl.totalActiveOffers.sub(1) (REX_dex.sol#235)
- gl.totalActiveOffers = 0 (REX_dex.sol#235)

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

Reentrancy in RexDEX.buyStakeFromList(uint256) (REX_dex.sol#208-242):

External calls:

- require(bool,string)(BUSD_TOKEN.transferFrom(msg.sender,MARKETING_ADDR,_fee),DEX: M-Fee transfer failed.) (REX_dex.sol#228)
- require(bool,string)(BUSD_TOKEN.transferFrom(msg.sender,DEVELOPMENT_ADDR,_fee),DEX: D-Fee transfer failed.) (REX_dex.sol#229)
- require(bool,string)(BUSD_TOKEN.transferFrom(msg.sender,_offer.staker,_offer.offerPrice),DEX: Transfer of BUSD failed.) (REX_dex.sol#233)
- REX_CONTRACT.createBoughtStake(_offer.stakeID,_offer.staker,msg.sender) (REX_dex.sol#239)

Event emitted after the call(s):

- StakeSold(_offer.staker,msg.sender,_offer.offerPrice,_offer.stakeID) (REX_dex.sol#241)

Reentrancy in RexDEX.revokeOffer(uint256) (REX_dex.sol#244-262):

External calls:

- REX_CONTRACT.restoreStake(msg.sender,_offer.stakeID) (REX_dex.sol#259)

Event emitted after the call(s):

- OfferRevoked(msg.sender,_offer.stakeID) (REX_dex.sol#261)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

RexDEX._notContract(address) (REX_dex.sol#272-273) uses assembly

- INLINE ASM (REX_dex.sol#273)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

RexSafeMath.mod(uint256,uint256) (REX_dex.sol#331-334) is never used and should be removed

RexSafeMath.mul(uint256,uint256) (REX_dex.sol#313-323) is never used and should be removed

RexSafeMath32.div(uint32,uint32) (REX_dex.sol#363-367) is never used and should be removed

RexSafeMath32.mod(uint32,uint32) (REX_dex.sol#369-372) is never used and should be removed

RexSafeMath32.mul(uint32,uint32) (REX_dex.sol#351-361) is never used and should be removed

RexSafeMath32.sub(uint32,uint32) (REX_dex.sol#345-349) is never used and should be removed

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.7.4 (REX_dex.sol#3) allows old versions

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Function IREXToken._checkStakeDataByID(address,bytes16) (REX_dex.sol#87-91) is not in mixedCase

Parameter RexDEX.initRexContract(address)._REX (REX_dex.sol#158) is not in mixedCase

Parameter RexDEX.listStake(address,uint32,uint32,uint256,bytes16)._staker (REX_dex.sol#173) is not in mixedCase

Parameter RexDEX.listStake(address,uint32,uint32,uint256,bytes16)._offerStartDay (REX_dex.sol#174) is not in mixedCase

Parameter RexDEX.listStake(address,uint32,uint32,uint256,bytes16)._offerDurationDays (REX_dex.sol#175) is not in mixedCase

Parameter RexDEX.listStake(address,uint32,uint32,uint256,bytes16)._offerPrice (REX_dex.sol#176) is not in mixedCase

Parameter RexDEX.listStake(address,uint32,uint32,uint256,bytes16)._stakeID (REX_dex.sol#177) is not in mixedCase

Parameter RexDEX.buyStakeFromList(uint256)._offerID (REX_dex.sol#209) is not in mixedCase

Parameter RexDEX.revokeOffer(uint256)._offerID (REX_dex.sol#245) is not in mixedCase

Function RexDEX._currentRxDay() (REX_dex.sol#268-270) is not in mixedCase

Parameter RexDEX.dataOfOfferNumber(uint256)._offerID (REX_dex.sol#275) is not in mixedCase

Variable RexDEX.TOKEN_DEFINER (REX_dex.sol#108) is not in mixedCase

Variable RexDEX.REX_CONTRACT (REX_dex.sol#109) is not in mixedCase

Variable RexDEX.BUSD_TOKEN (REX_dex.sol#110) is not in mixedCase

Variable RexDEX.MREX_TOKEN (REX_dex.sol#111) is not in mixedCase

Constant RexDEX.busd_address (REX_dex.sol#113) is not in UPPER_CASE_WITH_UNDERSCORES

Constant RexDEX.mrex_address (REX_dex.sol#116) is not in UPPER_CASE_WITH_UNDERSCORES

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

AIRDROP Contract:

Medium:

Contract locking ether found:

Contract RexAirdrop (REX_AIRDROP.sol#84-247) has payable functions:

- RexAirdrop.receive() (REX_AIRDROP.sol#116)
- RexAirdrop.fallback() (REX_AIRDROP.sol#117)

But does not have a function to withdraw the ether

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#contracts-that-lock-ether

Low & Informational:

Reentrancy in RexAirdrop.claimAirdropStake(uint32) (REX_AIRDROP.sol#206-232):

External calls:

- REX_CONTRACT.createStake(msg.sender,_amount,_stakingDays,0,true) (REX_AIRDROP.sol#230)

Event emitted after the call(s):

- AddressStakedDayAmountDuration(msg.sender,_currentRxDay(),_amount,_stakingDays) (REX_AIRDROP.sol#231)

Reentrancy in RexAirdrop.claimAirdropTokens() (REX_AIRDROP.sol#188-202):

External calls:

- REX_CONTRACT.mintSupply(msg.sender,_payout) (REX_AIRDROP.sol#200)

Event emitted after the call(s):

- AddressClaimedDayAmount(msg.sender,_currentRxDay(),_payout) (REX_AIRDROP.sol#201)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

RexAirdrop._notContract(address) (REX_AIRDROP.sol#242-246) uses assembly

- INLINE ASM (REX_AIRDROP.sol#244)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

RexAirdrop.initClaimables(address[],uint256[]) (REX_AIRDROP.sol#132-144) has costly operations inside a loop:

- totalAirdropAdresses = totalAirdropAdresses + 1 (REX_AIRDROP.sol#140)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop

RexSafeMath.mod(uint256,uint256) (REX_AIRDROP.sol#281-284) is never used and should be removed

RexSafeMath.sub(uint256,uint256) (REX_AIRDROP.sol#257-261) is never used and should be removed

RexSafeMath32.add(uint32,uint32) (REX_AIRDROP.sol#289-293) is never used and should be removed

RexSafeMath32.div(uint32,uint32) (REX_AIRDROP.sol#313-317) is never used and should be removed

RexSafeMath32.mod(uint32,uint32) (REX_AIRDROP.sol#319-322) is never used and should be removed

RexSafeMath32.mul(uint32,uint32) (REX_AIRDROP.sol#301-311) is never used and should be removed

RexSafeMath32.sub(uint32,uint32) (REX_AIRDROP.sol#295-299) is never used and should be removed

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.7.4 (REX_AIRDROP.sol#3) allows old versions

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Parameter RexAirdrop.initContract(address)._rex (REX_AIRDROP.sol#123) is not in mixedCase

Parameter RexAirdrop.initClaimables(address[],uint256[])._address (REX_AIRDROP.sol#133) is not in mixedCase

Parameter RexAirdrop.initClaimables(address[],uint256[])._amount (REX_AIRDROP.sol#134) is not in mixedCase

Parameter RexAirdrop.addressCanClaimNow(address)._claimer (REX_AIRDROP.sol#152) is not in mixedCase

Parameter RexAirdrop.addressCanStakeNow(address)._check (REX_AIRDROP.sol#165) is not in mixedCase

Parameter RexAirdrop.addressIsInAirdropList(address)._check (REX_AIRDROP.sol#178) is not in mixedCase

Parameter RexAirdrop.claimAirdropStake(uint32)._stakingDays (REX_AIRDROP.sol#207) is not in mixedCase

Function RexAirdrop._currentRxDay() (REX_AIRDROP.sol#238-240) is not in mixedCase

Variable RexAirdrop.TOKEN_DEFINER (REX_AIRDROP.sol#89) is not in mixedCase

Variable RexAirdrop.REX_CONTRACT (REX_AIRDROP.sol#91) is not in mixedCase

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

RexAirdrop.addressCanStakeNow(address) (REX_AIRDROP.sol#165-174) uses literals with too many digits:

- claimableAmount[_check] >= 1000000 && ! addressHasClaimed[_check] && _currentRxDay() >= 2 && _currentRxDay() <= LAST_CLAIM_DAY (REX_AIRDROP.sol#170-173)

RexAirdrop.claimAirdropStake(uint32) (REX_AIRDROP.sol#206-232) uses literals with too many digits:

- PERCENT_MORE_x_1E16 = (uint256(_stakingDaysCapped).mul(701754385964912)).add(78947368421100000) (REX_AIRDROP.sol#226)

RexAirdrop.claimAirdropStake(uint32) (REX_AIRDROP.sol#206-232) uses literals with too many digits:

- require(bool,string)(_amount >= 1000000,REX: Stake too small.) (REX_AIRDROP.sol#228)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

TREX Contract:

High:

TREX.buyOneTREX() (REX_TREX.sol#801-828) ignores return value by
BUSD_TOKEN.transfer(MARKETING_ADDR,price) (REX_TREX.sol#811)
TREX.buyOneTREX() (REX_TREX.sol#801-828) ignores return value by
BUSD_TOKEN.transfer(MARKETING_ADDR,marketingAndDevFundBUSD) (REX_TREX.sol#816)
TREX.buyOneTREX() (REX_TREX.sol#801-828) ignores return value by
BUSD_TOKEN.transfer(DEVELOPMENT_ADDR,marketingAndDevFundBUSD) (REX_TREX.sol#817)
TREX.sellAirdroppedTREX() (REX_TREX.sol#833-849) ignores return value by BUSD_TOKEN.transfer(msg.sender,price)
(REX_TREX.sol#846)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

Medium:

TREX.noOfTREXSellable(address) (REX_TREX.sol#854-868) performs a multiplication on the result of a division:
-
((_airdropBalances[_seller].div(10)).mul(_daysFromStart().div(SELL_EVERY_X_DAYS).add(1)).sub(_airdropBalances[_seller
].sub(_airdropBalancesLeft[_seller]))) > _airdropBalancesLeft[_seller] (REX_TREX.sol#855-867)
TREX.noOfTREXSellable(address) (REX_TREX.sol#854-868) performs a multiplication on the result of a division:
-
((_airdropBalances[_seller].div(10)).mul(_daysFromStart().div(SELL_EVERY_X_DAYS).add(1)).sub(_airdropBalances[_seller
].sub(_airdropBalancesLeft[_seller]))) (REX_TREX.sol#855-867)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

Contract locking ether found:
Contract TREX (REX_TREX.sol#720-902) has payable functions:
- TREX.receive() (REX_TREX.sol#899)
- TREX.fallback() (REX_TREX.sol#900)
But does not have a function to withdraw the ether
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#contracts-that-lock-ether

Reentrancy in TREX.buyOneTREX() (REX_TREX.sol#801-828):
External calls:
- require(bool,string)(BUSD_TOKEN.transferFrom(msg.sender,address(this),price),TREX: Transfer of BUSD failed.)
(REX_TREX.sol#803)
- BUSD_TOKEN.transfer(MARKETING_ADDR,price) (REX_TREX.sol#811)
- BUSD_TOKEN.transfer(MARKETING_ADDR,marketingAndDevFundBUSD) (REX_TREX.sol#816)
- BUSD_TOKEN.transfer(DEVELOPMENT_ADDR,marketingAndDevFundBUSD) (REX_TREX.sol#817)
State variables written after the call(s):
- price = price.add(priceRise) (REX_TREX.sol#821)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

Low & Informational:

BEP20Token.allowance(address,address).owner (REX_TREX.sol#542) shadows:

- Ownable.owner() (REX_TREX.sol#423-425) (function)

BEP20Token._approve(address,address,uint256).owner (REX_TREX.sol#697) shadows:

- Ownable.owner() (REX_TREX.sol#423-425) (function)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

TREX.initClaimables(address[],uint32[]) (REX_TREX.sol#764-777) should emit an event for:

- unSoldAirdropTREXes = unSoldAirdropTREXes.add(_amount[i]) (REX_TREX.sol#773)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

Reentrancy in TREX.buyOneTREX() (REX_TREX.sol#801-828):

External calls:

- require(bool,string)(BUSD_TOKEN.transferFrom(msg.sender,address(this),price),TREX: Transfer of BUSD failed.) (REX_TREX.sol#803)

State variables written after the call(s):

- soldTREX = soldTREX.add(1) (REX_TREX.sol#806)

Reentrancy in TREX.buyOneTREX() (REX_TREX.sol#801-828):

External calls:

- require(bool,string)(BUSD_TOKEN.transferFrom(msg.sender,address(this),price),TREX: Transfer of BUSD failed.) (REX_TREX.sol#803)
- BUSD_TOKEN.transfer(MARKETING_ADDR,price) (REX_TREX.sol#811)
- BUSD_TOKEN.transfer(MARKETING_ADDR,marketingAndDevFundBUSD) (REX_TREX.sol#816)
- BUSD_TOKEN.transfer(DEVELOPMENT_ADDR,marketingAndDevFundBUSD) (REX_TREX.sol#817)

State variables written after the call(s):

- _mint(msg.sender,1) (REX_TREX.sol#825)
  - _balances[account] = _balances[account].add(amount) (REX_TREX.sol#661)
- _mint(msg.sender,1) (REX_TREX.sol#825)
  - _totalSupply = _totalSupply.add(amount) (REX_TREX.sol#660)
- halvingNumber = halvingNumber.add(1) (REX_TREX.sol#822)

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

Reentrancy in TREX.buyOneTREX() (REX_TREX.sol#801-828):

External calls:

- require(bool,string)(BUSD_TOKEN.transferFrom(msg.sender,address(this),price),TREX: Transfer of BUSD failed.) (REX_TREX.sol#803)
- BUSD_TOKEN.transfer(MARKETING_ADDR,price) (REX_TREX.sol#811)
- BUSD_TOKEN.transfer(MARKETING_ADDR,marketingAndDevFundBUSD) (REX_TREX.sol#816)
- BUSD_TOKEN.transfer(DEVELOPMENT_ADDR,marketingAndDevFundBUSD) (REX_TREX.sol#817)

Event emitted after the call(s):

- ReceivedTREX(msg.sender,1) (REX_TREX.sol#827)
- Transfer(address(0),account,amount) (REX_TREX.sol#662)

- _mint(msg.sender,1) (REX_TREX.sol#825)

Reentrancy in TREX.sellAirdroppedTREX() (REX_TREX.sol#833-849):

External calls:

- BUSD_TOKEN.transfer(msg.sender,price) (REX_TREX.sol#846)

Event emitted after the call(s):

- SoldAirdroppedTREX(msg.sender,price) (REX_TREX.sol#848)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

TREX.sellAirdroppedTREX() (REX_TREX.sol#833-849) uses timestamp for comparisons

Dangerous comparisons:

- require(bool,string)(noOfTREXSellable(msg.sender) > 0,TREX: Address cannot sell right now.) (REX_TREX.sol#839)

TREX.noOfTREXSellable(address) (REX_TREX.sol#854-868) uses timestamp for comparisons

Dangerous comparisons:

-

((_airdropBalances[_seller].div(10)).mul(_daysFromStart().div(SELL_EVERY_X_DAYS).add(1)).sub(_airdropBalances[_seller].sub(_airdropBalancesLeft[_seller]))) > _airdropBalancesLeft[_seller] (REX_TREX.sol#855-867)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Address.isContract(address) (REX_TREX.sol#236-247) uses assembly

- INLINE ASM (REX_TREX.sol#243-245)

Address._functionCallWithValue(address,bytes,uint256,string) (REX_TREX.sol#344-370) uses assembly

- INLINE ASM (REX_TREX.sol#362-365)

TREX.isContract(address) (REX_TREX.sol#779-783) uses assembly

- INLINE ASM (REX_TREX.sol#781)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

TREX.initClaimables(address[],uint32[]) (REX_TREX.sol#764-777) has costly operations inside a loop:

- totalAirdropAdresses = totalAirdropAdresses.add(1) (REX_TREX.sol#772)

TREX.initClaimables(address[],uint32[]) (REX_TREX.sol#764-777) has costly operations inside a loop:

- unSoldAirdropTREXes = unSoldAirdropTREXes.add(_amount[i]) (REX_TREX.sol#773)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop

Address._functionCallWithValue(address,bytes,uint256,string) (REX_TREX.sol#344-370) is never used and should be removed

Address.functionCall(address,bytes) (REX_TREX.sol#291-293) is never used and should be removed

Address.functionCall(address,bytes,string) (REX_TREX.sol#301-307) is never used and should be removed

Address.functionCallWithValue(address,bytes,uint256) (REX_TREX.sol#320-326) is never used and should be removed

Address.functionCallWithValue(address,bytes,uint256,string) (REX_TREX.sol#334-342) is never used and should be removed

Address.isContract(address) (REX_TREX.sol#236-247) is never used and should be removed

Address.sendValue(address,uint256) (REX_TREX.sol#265-271) is never used and should be removed

BEP20Token._beforeTokenTransfer(address,address,uint256) (REX_TREX.sol#705) is never used and should be removed

SafeMath.mod(uint256,uint256) (REX_TREX.sol#156-158) is never used and should be removed

SafeMath.mod(uint256,uint256,string) (REX_TREX.sol#172-175) is never used and should be removed

SafeMath32.mod(uint32,uint32) (REX_TREX.sol#209-212) is never used and should be removed

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.7.4 (REX_TREX.sol#3) allows old versions

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (REX_TREX.sol#265-271):

- (success) = recipient.call{value: amount}() (REX_TREX.sol#269)

Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (REX_TREX.sol#344-370):

- (success,returndata) = target.call{value: weiValue}(data) (REX_TREX.sol#353)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter TREX.initClaimables(address[],uint32[])._address (REX_TREX.sol#765) is not in mixedCase

Parameter TREX.initClaimables(address[],uint32[])._amount (REX_TREX.sol#766) is not in mixedCase

Parameter TREX.noOfTREXSellable(address)._seller (REX_TREX.sol#854) is not in mixedCase

Parameter TREX.canClaimAirdropNow(address)._claimer (REX_TREX.sol#879) is not in mixedCase

Variable TREX.LAUNCH_TIME (REX_TREX.sol#732) is not in mixedCase

Constant TREX.priceRise (REX_TREX.sol#734) is not in UPPER_CASE_WITH_UNDERSCORES

Variable TREX.BUSD_TOKEN (REX_TREX.sol#741) is not in mixedCase

Constant TREX.busd_address (REX_TREX.sol#743) is not in UPPER_CASE_WITH_UNDERSCORES

Variable TREX._airdropBalances (REX_TREX.sol#747) is not in mixedCase

Variable TREX._airdropBalancesLeft (REX_TREX.sol#748) is not in mixedCase

Variable TREX._airdropClaimed (REX_TREX.sol#749) is not in mixedCase

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

REX Contract:

Medium:

StakingToken.endStake(bytes16) (REX_REX.sol#811-888) performs a multiplication on the result of a division:

-stakesSharesCorr = stakesSharesCorr.mul(80).div(100) (REX_REX.sol#869)

-stakesSharesCorr = initialShares[msg.sender][_stakeID].mul(80).div(100) (REX_REX.sol#864-866)

StakingToken._stakesShares(uint256,uint32,uint256) (REX_REX.sol#970-984) performs a multiplication on the result of a division:

-_stakedAmount.mul(PRECISION_RATE).div(_sharePrice).mul(SHARES_PRECISION + _getBonus(_stakingDays)).div(SHARES_PRECISION) (REX_REX.sol#979-983)

StakingToken._getBonus(uint32) (REX_REX.sol#986-1002) performs a multiplication on the result of a division:

-fullYears = _stakingDays.div(365) (REX_REX.sol#994)

-_days += (_stakingDays - 365 * fullYears) * (fullYears + 1) (REX_REX.sol#999)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

StakingToken._calculatePenaltyAmount(Declaration.Stake) (REX_REX.sol#1120-1129) uses a dangerous strict equality:
- (_stakeNotStarted(_stake) || _daysLeft(_stake) == 0) (REX_REX.sol#1127-1128)
StakingToken._checkPenaltyAmountbyID(address,bytes16) (REX_REX.sol#1105-1112) uses a dangerous strict equality:
- stake.isActive == 0 (REX_REX.sol#1107-1111)
StakingToken._checkPenaltyAmountbyID(address,bytes16) (REX_REX.sol#1105-1112) uses a dangerous strict equality:
- stake.isActive == 1 || stake.isActive == 2 (REX_REX.sol#1107-1111)
StakingToken._checkPenaltyAmountbyID(address,bytes16) (REX_REX.sol#1105-1112) uses a dangerous strict equality:
- stake.isActive == 3 || stake.isActive == 4 (REX_REX.sol#1107-1111)
StakingToken._checkRewardAmountbyID(address,bytes16,uint32) (REX_REX.sol#1015-1097) uses a dangerous strict equality:
- stake.isActive == 0 (REX_REX.sol#1026)
StakingToken._checkRewardAmountbyID(address,bytes16,uint32) (REX_REX.sol#1015-1097) uses a dangerous strict equality:
- stake.isActive == 3 || stake.isActive == 4 (REX_REX.sol#1028)
StakingToken._checkRewardAmountbyID(address,bytes16,uint32) (REX_REX.sol#1015-1097) uses a dangerous strict equality:
- stake.isIrrTrex == 2 (REX_REX.sol#1084-1088)
Helper._daysLeft(Declaration.Stake) (REX_REX.sol#581-585) uses a dangerous strict equality:
- _stake.isActive == 0 (REX_REX.sol#582-584)
Helper._startingDay(Declaration.Stake) (REX_REX.sol#591-593) uses a dangerous strict equality:
- _stake.withdrawDay == 0 (REX_REX.sol#592)
StakingToken.endStake(bytes16) (REX_REX.sol#811-888) uses a dangerous strict equality:
- _stake.isIrrTrex == 1 || _stake.isIrrTrex == 3 (REX_REX.sol#868)
StakingToken.endStake(bytes16) (REX_REX.sol#811-888) uses a dangerous strict equality:
- (_stake.isIrrTrex == 2 || _stake.isIrrTrex == 3) (REX_REX.sol#864-866)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Contract locking ether found:
Contract RexToken (REX_REX.sol#1439-1496) has payable functions:
- RexToken.receive() (REX_REX.sol#1448)
- RexToken.fallback() (REX_REX.sol#1449)
But does not have a function to withdraw the ether
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#contracts-that-lock-ether

Helper.stakesPagination(address,uint256,uint256).i (REX_REX.sol#559) is a local variable never initialized
DexToken.createBoughtStake(bytes16,address,address)._newStake (REX_REX.sol#1409) is a local variable never initialized
ExtendedStaking.transferStake(bytes16,address)._newStake (REX_REX.sol#1174) is a local variable never initialized
ExtendedStaking.splitStake(bytes16,uint256)._newStake (REX_REX.sol#1246) is a local variable never initialized

StakingToken.createStake(address,uint256,uint32,string,bool)._newStake (REX_REX.sol#733) is a local variable never initialized

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

Low & Informational:

Snapshot._inflationAmount(uint256,uint256,uint256)._totalSupply (REX_REX.sol#676) shadows:

- BEP20Token._totalSupply (REX_REX.sol#246) (state variable)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

RexToken.initRexContracts(address,address,address,address)._AIRDROP (REX_REX.sol#1455) lacks a zero-check on :

- AIRDROP_CONTRACT = _AIRDROP (REX_REX.sol#1457)

RexToken.initRexContracts(address,address,address,address)._RDA (REX_REX.sol#1455) lacks a zero-check on :

- RDA_CONTRACT = _RDA (REX_REX.sol#1458)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

StakingToken.createStake(address,uint256,uint32,string,bool) (REX_REX.sol#722-805) has external calls inside a loop: (TREX_TOKEN.balanceOf(_staker) > 0) (REX_REX.sol#758-760)

StakingToken.createStake(address,uint256,uint32,string,bool) (REX_REX.sol#722-805) has external calls inside a loop: TREX_TOKEN.balanceOf(_staker) > 0 (REX_REX.sol#765)

StakingToken.createStake(address,uint256,uint32,string,bool) (REX_REX.sol#722-805) has external calls inside a loop: TREX_TOKEN.balanceOf(_staker) > 0 (REX_REX.sol#778)

StakingToken.createStake(address,uint256,uint32,string,bool) (REX_REX.sol#722-805) has external calls inside a loop: TREX_TOKEN.balanceOf(_staker) > 0 (REX_REX.sol#782)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Timing.currentRxDay() (REX_REX.sol#506-508) uses timestamp for comparisons

Dangerous comparisons:

- block.timestamp >= LAUNCH_TIME (REX_REX.sol#507)

Helper._stakeNotStarted(Declaration.Stake) (REX_REX.sol#571-575) uses timestamp for comparisons

Dangerous comparisons:

- _stake.startDay > _currentRxDay() (REX_REX.sol#572-574)

Helper._daysDiff(uint32,uint32) (REX_REX.sol#577-579) uses timestamp for comparisons

Dangerous comparisons:

- _startDate > _endDate (REX_REX.sol#578)

Snapshot.manualSnapshotOneDay() (REX_REX.sol#615-623) uses timestamp for comparisons

Dangerous comparisons:

- currentRxDay() >= 2 (REX_REX.sol#620)
- currentRxDay() > globals.currentRxDay + 1 (REX_REX.sol#621)

StakingToken.endStake(bytes16) (REX_REX.sol#811-888) uses timestamp for comparisons

Dangerous comparisons:

- require(bool,string)(stakes[msg.sender][_stakeID].isActive == 1,RX: 4) (REX_REX.sol#818)
- require(bool,string)(stakes[msg.sender][_stakeID].finalDay <= _currentRxDay(),RX: 6) (REX_REX.sol#823)
- _stake.penaltyAmount > 0 (REX_REX.sol#856)
- _stake.isIrrTrex == 1 || _stake.isIrrTrex == 3 (REX_REX.sol#868)
- totalREXinActiveStakes[msg.sender] >= (_stake.stakedAmount) (REX_REX.sol#834-836)
- _stake.stakedAmount > _stake.penaltyAmount (REX_REX.sol#839-843)
- (_stake.isIrrTrex == 2 || _stake.isIrrTrex == 3) (REX_REX.sol#864-866)
- _stake.stakedAmount > _stake.penaltyAmount (REX_REX.sol#872-877)

StakingToken._removeScheduledShares(uint32,uint256) (REX_REX.sol#900-919) uses timestamp for comparisons

Dangerous comparisons:

- _finalDay >= _currentRxDay() (REX_REX.sol#906)
- scheduledToEnd[_finalDay] > _shares (REX_REX.sol#908-910)
- snapshots[_day].scheduledToEnd > _shares (REX_REX.sol#915-917)

StakingToken._sharePriceUpdate(uint256,uint256,uint32,uint256) (REX_REX.sol#921-952) uses timestamp for comparisons

Dangerous comparisons:

- _stakeShares > 0 && _currentRxDay() > 3 (REX_REX.sol#929)
- newSharePrice > globals.sharePrice (REX_REX.sol#937)
- newSharePrice < globals.sharePrice.mul(105).div(100) (REX_REX.sol#939-941)

StakingToken._getBonus(uint32) (REX_REX.sol#986-1002) uses timestamp for comparisons

Dangerous comparisons:

- i <= fullYears (REX_REX.sol#995)

StakingToken._checkRewardAmountbyID(address,bytes16,uint32) (REX_REX.sol#1015-1097) uses timestamp for comparisons

Dangerous comparisons:

- stake.isActive == 0 (REX_REX.sol#1026)
- stake.startDay > _currentRxDay() (REX_REX.sol#1027)
- stake.isActive == 3 || stake.isActive == 4 (REX_REX.sol#1028)
- _currentRxDay() >= stake.finalDay (REX_REX.sol#1030)
- _currentRxDay() > (_finalDay + uint32(14)) && rewardAmount > 0 (REX_REX.sol#1045)
- _reductionPercent > 100 (REX_REX.sol#1047)
- _withdrawDay <= _startingDay(stake) (REX_REX.sol#1067)
- rewardAmount > 0 (REX_REX.sol#1082)
- _withdrawDay > _currentRxDay() (REX_REX.sol#1063-1065)
- stake.isIrrTrex == 2 (REX_REX.sol#1084-1088)

StakingToken._checkPenaltyAmountbyID(address,bytes16) (REX_REX.sol#1105-1112) uses timestamp for comparisons

Dangerous comparisons:

- stake.isActive == 0 (REX_REX.sol#1107-1111)
- stake.isActive == 1 || stake.isActive == 2 (REX_REX.sol#1107-1111)
- stake.isActive == 3 || stake.isActive == 4 (REX_REX.sol#1107-1111)

StakingToken._calculatePenaltyAmount(Declaration.Stake) (REX_REX.sol#1120-1129) uses timestamp for comparisons

Dangerous comparisons:

- (_stakeNotStarted(_stake) || _daysLeft(_stake) == 0) (REX_REX.sol#1127-1128)

StakingToken._loopRewardAmount(uint256,uint32,uint32) (REX_REX.sol#1131-1144) uses timestamp for comparisons

Dangerous comparisons:

- _day < _finalDay (REX_REX.sol#1141)

ExtendedStaking.transferStake(bytes16,address) (REX_REX.sol#1159-1201) uses timestamp for comparisons

Dangerous comparisons:

- require(bool,string)(stakes[msg.sender][_stakeID].isActive == 1,RX: 7) (REX_REX.sol#1169)
- require(bool,string)(stakes[msg.sender][_stakeID].withdrawDay == 0,RX: 8) (REX_REX.sol#1170)

ExtendedStaking.renameStake(bytes16,string) (REX_REX.sol#1209-1221) uses timestamp for comparisons

Dangerous comparisons:

- require(bool,string)(stakes[msg.sender][_stakeID].isActive == 1,RX: 10) (REX_REX.sol#1217)

ExtendedStaking.splitStake(bytes16,uint256) (REX_REX.sol#1229-1276) uses timestamp for comparisons

Dangerous comparisons:

- require(bool,string)(stakes[msg.sender][_stakeID].isActive == 1,RX: 11) (REX_REX.sol#1237)
- require(bool,string)(! stakes[msg.sender][_stakeID].isSplit,RX: 12) (REX_REX.sol#1238)
- require(bool,string)(stakes[msg.sender][_stakeID].withdrawDay == 0,RX: 13) (REX_REX.sol#1239)

ExtendedStaking.withdrawRewards(bytes16,uint32) (REX_REX.sol#1283-1326) uses timestamp for comparisons

Dangerous comparisons:

- require(bool,string)(stakes[msg.sender][_stakeID].isActive == 1,RX: 16) (REX_REX.sol#1296)
- require(bool,string)(stakes[msg.sender][_stakeID].isIrrTrex == 0 || stakes[msg.sender][_stakeID].isIrrTrex == 2,RX: 18) (REX_REX.sol#1297)
- require(bool,string)(_currentRxDay() > stakes[msg.sender][_stakeID].startDay,RX: 17) (REX_REX.sol#1298)
- require(bool,string)(stakes[msg.sender][_stakeID].finalDay > _currentRxDay(),RX: 17a) (REX_REX.sol#1299)
- require(bool,string)(withdrawAmount > 0,RX: Fail) (REX_REX.sol#1304)
- stake.stakesShares > stakersPenalty (REX_REX.sol#1309-1311)
- stakersPenalty > _sharesTemp (REX_REX.sol#1314)
- _sharesTemp > stakersPenalty (REX_REX.sol#1315)

DexToken.offerStake(bytes16,uint256,uint32) (REX_REX.sol#1342-1369) uses timestamp for comparisons

Dangerous comparisons:

- require(bool,string)((_currentRxDay() + _durationDays) < stakes[msg.sender][_stakeID].finalDay,RX: 21) (REX_REX.sol#1354)
- require(bool,string)(_currentRxDay() >= ((stakes[msg.sender][_stakeID].stakingDays).mul(10).div(100).add(stakes[msg.sender][_stakeID].startDay)),RX: 22) (REX_REX.sol#1355)
- require(bool,string)(_currentRxDay() >= DEX_ACTIVATION_DAY,RX: 23) (REX_REX.sol#1356)
- require(bool,string)(stakes[msg.sender][_stakeID].isActive == 1,RX: 24) (REX_REX.sol#1359)
- require(bool,string)(stakes[msg.sender][_stakeID].withdrawDay == 0,RX: 25) (REX_REX.sol#1360)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Helper._notContract(address) (REX_REX.sol#524-528) uses assembly

- INLINE ASM (REX_REX.sol#526)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

BEP20Token._burn(address,uint256) (REX_REX.sol#333-338) has costly operations inside a loop:

- _totalSupply = _totalSupply.sub(amount) (REX_REX.sol#336)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop

SafeMath.mod(uint256,uint256) (REX_REX.sol#193-195) is never used and should be removed

SafeMath.mod(uint256,uint256,string) (REX_REX.sol#197-200) is never used and should be removed

SafeMath32.mod(uint32,uint32) (REX_REX.sol#234-237) is never used and should be removed

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.7.4 (REX_REX.sol#3) allows old versions

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Constant BEP20Token._decimals (REX_REX.sol#247) is not in UPPER_CASE_WITH_UNDERSCORES

Variable Declaration.LAUNCH_TIME (REX_REX.sol#459) is not in mixedCase

Constant Declaration.busd_address (REX_REX.sol#467) is not in UPPER_CASE_WITH_UNDERSCORES

Variable Declaration.RDA_CONTRACT (REX_REX.sol#468) is not in mixedCase

Variable Declaration.AIRDROP_CONTRACT (REX_REX.sol#469) is not in mixedCase

Variable Declaration.TREX_TOKEN (REX_REX.sol#472) is not in mixedCase

Variable Declaration.DEX_CONTRACT (REX_REX.sol#473) is not in mixedCase

Parameter Helper.stakesPagination(address,uint256,uint256)._staker (REX_REX.sol#543) is not in mixedCase

Parameter Helper.stakesPagination(address,uint256,uint256)._offset (REX_REX.sol#544) is not in mixedCase

Parameter Helper.stakesPagination(address,uint256,uint256)._length (REX_REX.sol#545) is not in mixedCase

Parameter StakingToken.createStakeBatch(uint256[],uint32[],string,bool[])._stakedAmount (REX_REX.sol#695) is not in mixedCase

Parameter StakingToken.createStakeBatch(uint256[],uint32[],string,bool[])._stakingDays (REX_REX.sol#696) is not in mixedCase

Parameter StakingToken.createStakeBatch(uint256[],uint32[],string,bool[])._description (REX_REX.sol#697) is not in mixedCase

Parameter StakingToken.createStakeBatch(uint256[],uint32[],string,bool[])._irrevocable (REX_REX.sol#698) is not in mixedCase

Parameter StakingToken.createStake(address,uint256,uint32,string,bool)._stakerAdd (REX_REX.sol#723) is not in mixedCase

Parameter StakingToken.createStake(address,uint256,uint32,string,bool)._stakedAmount (REX_REX.sol#724) is not in mixedCase

Parameter StakingToken.createStake(address,uint256,uint32,string,bool)._stakingDays (REX_REX.sol#725) is not in mixedCase

Parameter StakingToken.createStake(address,uint256,uint32,string,bool)._description (REX_REX.sol#726) is not in mixedCase

Parameter StakingToken.createStake(address,uint256,uint32,string,bool)._irrevocable (REX_REX.sol#727) is not in mixedCase

Parameter StakingToken.endStake(bytes16)._stakeID (REX_REX.sol#812) is not in mixedCase

Function StakingToken._checkStakeDataByID(address,bytes16) (REX_REX.sol#1004-1006) is not in mixedCase

Parameter StakingToken._checkStakeDataByID(address,bytes16)._staker (REX_REX.sol#1004) is not in mixedCase

Parameter StakingToken._checkStakeDataByID(address,bytes16)._stakeID (REX_REX.sol#1004) is not in mixedCase

Function StakingToken._checkRewardAmountbyID(address,bytes16,uint32) (REX_REX.sol#1015-1097) is not in mixedCase

Parameter StakingToken._checkRewardAmountbyID(address,bytes16,uint32)._staker (REX_REX.sol#1016) is not in mixedCase

Parameter StakingToken._checkRewardAmountbyID(address,bytes16,uint32)._stakeID (REX_REX.sol#1017) is not in mixedCase

Parameter StakingToken._checkRewardAmountbyID(address,bytes16,uint32)._withdrawDays (REX_REX.sol#1018) is not in mixedCase

Function StakingToken._checkPenaltyAmountbyID(address,bytes16) (REX_REX.sol#1105-1112) is not in mixedCase

Parameter StakingToken._checkPenaltyAmountbyID(address,bytes16)._staker (REX_REX.sol#1105) is not in mixedCase

Parameter StakingToken._checkPenaltyAmountbyID(address,bytes16)._stakeID (REX_REX.sol#1105) is not in mixedCase

Parameter ExtendedStaking.transferStake(bytes16,address)._stakeID (REX_REX.sol#1160) is not in mixedCase

Parameter ExtendedStaking.transferStake(bytes16,address)._toAddress (REX_REX.sol#1161) is not in mixedCase

Parameter ExtendedStaking.renameStake(bytes16,string)._stakeID (REX_REX.sol#1210) is not in mixedCase

Parameter ExtendedStaking.renameStake(bytes16,string)._description (REX_REX.sol#1211) is not in mixedCase

Parameter ExtendedStaking.splitStake(bytes16,uint256)._stakeID (REX_REX.sol#1230) is not in mixedCase

Parameter ExtendedStaking.withdrawRewards(bytes16,uint32)._stakeID (REX_REX.sol#1284) is not in mixedCase

Parameter ExtendedStaking.withdrawRewards(bytes16,uint32)._withdrawDays (REX_REX.sol#1285) is not in mixedCase

Parameter DexToken.offerStake(bytes16,uint256,uint32)._stakeID (REX_REX.sol#1343) is not in mixedCase

Parameter DexToken.offerStake(bytes16,uint256,uint32)._price (REX_REX.sol#1344) is not in mixedCase

Parameter DexToken.offerStake(bytes16,uint256,uint32)._durationDays (REX_REX.sol#1345) is not in mixedCase

Parameter DexToken.restoreStake(address,bytes16)._staker (REX_REX.sol#1376) is not in mixedCase

Parameter DexToken.restoreStake(address,bytes16)._stakeID (REX_REX.sol#1377) is not in mixedCase

Parameter DexToken.createBoughtStake(bytes16,address,address)._stakeID (REX_REX.sol#1398) is not in mixedCase

Parameter DexToken.createBoughtStake(bytes16,address,address)._fromAddress (REX_REX.sol#1399) is not in mixedCase

Parameter DexToken.createBoughtStake(bytes16,address,address)._toAddress (REX_REX.sol#1400) is not in mixedCase

Parameter RexToken.initRexContracts(address,address,address,address)._AIRDROP (REX_REX.sol#1455) is not in mixedCase

Parameter RexToken.initRexContracts(address,address,address,address)._RDA (REX_REX.sol#1455) is not in mixedCase

Parameter RexToken.initRexContracts(address,address,address,address)._DEX (REX_REX.sol#1455) is not in mixedCase

Parameter RexToken.initRexContracts(address,address,address,address)._TREX (REX_REX.sol#1455) is not in mixedCase

Parameter RexToken.mintSupply(address,uint256)._donatorAddress (REX_REX.sol#1478) is not in mixedCase

Parameter RexToken.mintSupply(address,uint256)._amount (REX_REX.sol#1479) is not in mixedCase

Variable RexToken.TOKEN_DEFINER (REX_REX.sol#1441) is not in mixedCase

Variable RexToken.UNISWAP_PAIR (REX_REX.sol#1442) is not in mixedCase

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

RexToken.slitherConstructorConstantVariables() (REX_REX.sol#1439-1496) uses literals with too many digits:

- MIN_STAKE_AMOUNT = 1000000 (REX_REX.sol#463)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

RDA Contract:

High:

RexDailyAuction._generateSupplyAndCheckPools(uint32) (REX_Rda.sol#607-647) sends eth to arbitrary user

Dangerous calls:

- msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations

RexDailyAuction._getRandomNumber(uint256) (REX_Rda.sol#855-864) uses a weak PRNG: "val = val % uint256(ceiling) (REX_Rda.sol#860)"

RexDailyAuction._getAnotherRandomNumber(uint256) (REX_Rda.sol#866-875) uses a weak PRNG: "val = val % uint256(ceiling) (REX_Rda.sol#871)"

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG

Reentrancy in RexDailyAuction.claimBusdFromBPD() (REX_Rda.sol#1176-1188):

External calls:

- supplyTrigger() (REX_Rda.sol#1177)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#828)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#829)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_startLiquidity) (REX_Rda.sol#830)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_startLiquidity,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#832-846)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#724)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#725)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount) (REX_Rda.sol#726)
  - BUSD_TOKEN.transfer(MARKETING_ADDR,amo) (REX_Rda.sol#634)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_busdAmount,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#728-742)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount_scope_3) (REX_Rda.sol#776)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount_scope_3) (REX_Rda.sol#777)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount_scope_2) (REX_Rda.sol#778)

- (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount_scope_3,_busdAmount_scope_2,0,0,address(0x0),block.timestamp.add(7200)) (REX_Rda.sol#780-794)

External calls sending eth:

- supplyTrigger() (REX_Rda.sol#1177)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

State variables written after the call(s):

- g.totalClaimedRandomBUSD = g.totalClaimedRandomBUSD.add(claimed) (REX_Rda.sol#1185)
- randomBUSD[msg.sender] = 0 (REX_Rda.sol#1184)

Reentrancy in RexDailyAuction.claimBusdFromReferrals() (REX_Rda.sol#1159-1171):

External calls:

- supplyTrigger() (REX_Rda.sol#1160)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#828)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#829)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_startLiquidity) (REX_Rda.sol#830)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_startLiquidity,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#832-846)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#724)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#725)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount) (REX_Rda.sol#726)
  - BUSD_TOKEN.transfer(MARKETING_ADDR,amo) (REX_Rda.sol#634)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_busdAmount,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#728-742)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount_scope_3) (REX_Rda.sol#776)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount_scope_3) (REX_Rda.sol#777)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount_scope_2) (REX_Rda.sol#778)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount_scope_3,_busdAmount_scope_2,0,0,address(0x0),block.timestamp.add(7200)) (REX_Rda.sol#780-794)

External calls sending eth:

- supplyTrigger() (REX_Rda.sol#1160)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

State variables written after the call(s):

- g.totalClaimedReferralBUSD = g.totalClaimedReferralBUSD.add(claimed) (REX_Rda.sol#1168)

Reentrancy in RexDailyAuction.claimBusdFromTREASURY() (REX_Rda.sol#1193-1208):

External calls:

- supplyTrigger() (REX_Rda.sol#1194)

- (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
- REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#828)
- REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#829)
- BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_startLiquidity) (REX_Rda.sol#830)
- (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_startLiquidity,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#832-846)
- REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#724)
- REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#725)
- BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount) (REX_Rda.sol#726)
- BUSD_TOKEN.transfer(MARKETING_ADDR,amo) (REX_Rda.sol#634)
- (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_busdAmount,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#728-742)
- REX_CONTRACT.mintSupply(address(this),_rexAmount_scope_3) (REX_Rda.sol#776)
- REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount_scope_3) (REX_Rda.sol#777)
- BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount_scope_2) (REX_Rda.sol#778)
- (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount_scope_3,_busdAmount_scope_2,0,0,address(0x0),block.timestamp.add(7200)) (REX_Rda.sol#780-794)

External calls sending eth:

- supplyTrigger() (REX_Rda.sol#1194)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

State variables written after the call(s):

- BUSDTREASURY = BUSDTREASURY.sub(claimed) (REX_Rda.sol#1206)
- addressHitByRandom[msg.sender] = true (REX_Rda.sol#1205)

Reentrancy in RexDailyAuction.claimRexFromDonations() (REX_Rda.sol#1102-1127):

External calls:

- supplyTrigger() (REX_Rda.sol#1103)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#828)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#829)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_startLiquidity) (REX_Rda.sol#830)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_startLiquidity,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#832-846)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#724)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#725)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount) (REX_Rda.sol#726)
  - BUSD_TOKEN.transfer(MARKETING_ADDR,amo) (REX_Rda.sol#634)

- (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_busdAmount,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#728-742)
- REX_CONTRACT.mintSupply(address(this),_rexAmount_scope_3) (REX_Rda.sol#776)
- REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount_scope_3) (REX_Rda.sol#777)
- BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount_scope_2) (REX_Rda.sol#778)
- (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount_scope_3,_busdAmount_scope_2,0,0,address(0x0),block.timestamp.add(7200)) (REX_Rda.sol#780-794)

External calls sending eth:

- supplyTrigger() (REX_Rda.sol#1103)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

State variables written after the call(s):

- g.totalClaimedDonationREX = g.totalClaimedDonationREX.add(_payout) (REX_Rda.sol#1122)
- _removeEligibleAddr(msg.sender) (REX_Rda.sol#1121)
  - userAddressesBPD[indexToDelete] = addressToMove (REX_Rda.sol#928)
  - userAddressesBPD.pop() (REX_Rda.sol#931)

Reentrancy in RexDailyAuction.claimRexFromReferrals() (REX_Rda.sol#1134-1154):

External calls:

- supplyTrigger() (REX_Rda.sol#1135)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#828)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#829)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_startLiquidity) (REX_Rda.sol#830)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_startLiquidity,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#832-846)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#724)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#725)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount) (REX_Rda.sol#726)
  - BUSD_TOKEN.transfer(MARKETING_ADDR,amo) (REX_Rda.sol#634)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_busdAmount,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#728-742)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount_scope_3) (REX_Rda.sol#776)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount_scope_3) (REX_Rda.sol#777)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount_scope_2) (REX_Rda.sol#778)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount_scope_3,_busdAmount_scope_2,0,0,address(0x0),block.timestamp.add(7200)) (REX_Rda.sol#780-794)

External calls sending eth:

- supplyTrigger() (REX_Rda.sol#1135)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

State variables written after the call(s):
- g.totalClaimedReferralREX = g.totalClaimedReferralREX.add(_payout) (REX_Rda.sol#1150)

Reentrancy in RexDailyAuction.claimStakeFromDonations(uint32) (REX_Rda.sol#1066-1094):

External calls:
- supplyTrigger() (REX_Rda.sol#1069)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#828)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#829)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_startLiquidity) (REX_Rda.sol#830)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_startLiquidity,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#832-846)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#724)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#725)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount) (REX_Rda.sol#726)
  - BUSD_TOKEN.transfer(MARKETING_ADDR,amo) (REX_Rda.sol#634)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_busdAmount,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#728-742)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount_scope_3) (REX_Rda.sol#776)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount_scope_3) (REX_Rda.sol#777)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount_scope_2) (REX_Rda.sol#778)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount_scope_3,_busdAmount_scope_2,0,0,address(0x0),block.timestamp.add(7200)) (REX_Rda.sol#780-794)

External calls sending eth:
- supplyTrigger() (REX_Rda.sol#1069)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

State variables written after the call(s):
- g.totalClaimedDonationREX = g.totalClaimedDonationREX.add(_payout) (REX_Rda.sol#1089)
- _addEligibleAddr(msg.sender) (REX_Rda.sol#1088)
  - userAddressesBPD.push(_userAddress) (REX_Rda.sol#900)
  - userAddressesBPD.push(userAddressesBPD[_pos]) (REX_Rda.sol#906)
  - userAddressesBPD[_pos] = _userAddress (REX_Rda.sol#908)

Reentrancy in RexDailyAuction.donateBUSD(uint256,address) (REX_Rda.sol#352-366):

External calls:
- require(bool,string)(BUSD_TOKEN.transferFrom(msg.sender,address(this),_busd_amount),REX: Transfer of BUSD failed.) (REX_Rda.sol#363)

- _reserveRex(_referralAddress,msg.sender,_busd_amount) (REX_Rda.sol#365)
  - BUSD_TOKEN.transfer(MARKETING_ADDR,_senderValue.div(20)) (REX_Rda.sol#410)
  - BUSD_TOKEN.transfer(DEVELOPMENT_ADDR,_senderValue.div(20)) (REX_Rda.sol#411)
- supplyTrigger() (REX_Rda.sol#353)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#828)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#829)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_startLiquidity) (REX_Rda.sol#830)
  - (amountREX,amountBUSD) =
UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_startLiquidity,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#832-846)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#724)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#725)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount) (REX_Rda.sol#726)
  - BUSD_TOKEN.transfer(MARKETING_ADDR,amo) (REX_Rda.sol#634)
  - (amountREX,amountBUSD) =
UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_busdAmount,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#728-742)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount_scope_3) (REX_Rda.sol#776)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount_scope_3) (REX_Rda.sol#777)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount_scope_2) (REX_Rda.sol#778)
  - (amountREX,amountBUSD) =
UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount_scope_3,_busdAmount_scope_2,0,0,address(0x0),block.timestamp.add(7200)) (REX_Rda.sol#780-794)

External calls sending eth:

- supplyTrigger() (REX_Rda.sol#353)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

State variables written after the call(s):

- _reserveRex(_referralAddress,msg.sender,_busd_amount) (REX_Rda.sol#365)
  - BUSDPOOL = BUSDPOOL.add(_senderValue.mul(75).div(100)) (REX_Rda.sol#408)
- _reserveRex(_referralAddress,msg.sender,_busd_amount) (REX_Rda.sol#365)
  - BUSDTREASURY = BUSDTREASURY.add(_senderValue.mul(uint256(5).sub(mrexRef)).div(100)) (REX_Rda.sol#426)
  - BUSDTREASURY = BUSDTREASURY.add(_senderValue.mul(5).div(100)) (REX_Rda.sol#432)
- _reserveRex(_referralAddress,msg.sender,_busd_amount) (REX_Rda.sol#365)
  - dailyTotalDonation[_donationDay] = dailyTotalDonation[_donationDay].add(_donationBalance) (REX_Rda.sol#452)
- _reserveRex(_referralAddress,msg.sender,_busd_amount) (REX_Rda.sol#365)
  - dailyTotalReferral[_donationDay] = dailyTotalReferral[_donationDay].add(_referralAmount) (REX_Rda.sol#465)
- _reserveRex(_referralAddress,msg.sender,_busd_amount) (REX_Rda.sol#365)
  - donatorBalances[_senderAddress][_donationDay] =
donatorBalances[_senderAddress][_donationDay].add(_donationBalance) (REX_Rda.sol#451)

- _reserveRex(_referralAddress,msg.sender,_busd_amount) (REX_Rda.sol#365)
  - g.totalDonatedBUSD = g.totalDonatedBUSD.add(_senderValue) (REX_Rda.sol#406)
- _reserveRex(_referralAddress,msg.sender,_busd_amount) (REX_Rda.sol#365)
  - originalDonation[_senderAddress] = originalDonation[_senderAddress].add(_senderValue) (REX_Rda.sol#405)
- _reserveRex(_referralAddress,msg.sender,_busd_amount) (REX_Rda.sol#365)
  - sumOfDonationsOfUnHit = sumOfDonationsOfUnHit.add(_senderValue) (REX_Rda.sol#400)
- _reserveRex(_referralAddress,msg.sender,_busd_amount) (REX_Rda.sol#365)
  - toSendToPairBusd = toSendToPairBusd.add(_senderValue.div(10)) (REX_Rda.sol#409)

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

RexDailyAuction._reserveRex(address,address,uint256) (REX_Rda.sol#373-434) ignores return value by BUSD_TOKEN.transfer(MARKETING_ADDR,_senderValue.div(20)) (REX_Rda.sol#410)

RexDailyAuction._reserveRex(address,address,uint256) (REX_Rda.sol#373-434) ignores return value by BUSD_TOKEN.transfer(DEVELOPMENT_ADDR,_senderValue.div(20)) (REX_Rda.sol#411)

RexDailyAuction.withdrawLPTokens() (REX_Rda.sol#521-534) ignores return value by LP_TOKEN.transfer(msg.sender,lpTokensPayout) (REX_Rda.sol#531)

RexDailyAuction._generateSupplyAndCheckPools(uint32) (REX_Rda.sol#607-647) ignores return value by BUSD_TOKEN.transfer(MARKETING_ADDR,amo) (REX_Rda.sol#634)

RexDailyAuction.claimBusdFromReferrals() (REX_Rda.sol#1159-1171) ignores return value by BUSD_TOKEN.transfer(msg.sender,claimed) (REX_Rda.sol#1169)

RexDailyAuction.claimBusdFromBPD() (REX_Rda.sol#1176-1188) ignores return value by BUSD_TOKEN.transfer(msg.sender,claimed) (REX_Rda.sol#1186)

RexDailyAuction.claimBusdFromTREASURY() (REX_Rda.sol#1193-1208) ignores return value by BUSD_TOKEN.transfer(msg.sender,claimed) (REX_Rda.sol#1207)

RexDailyAuction.withdrawTokensAfterContractEnd(address) (REX_Rda.sol#1318-1331) ignores return value by Token.transfer(MARKETING_ADDR,amo) (REX_Rda.sol#1329)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

Medium:

RexDailyAuction.withdrawableLPTokens(address) (REX_Rda.sol#539-551) performs a multiplication on the result of a division:

-lpTokenPayout = totalLpTokens.mul(liquidityBalances[who]).div(INITIAL_LIQ_BUSD) (REX_Rda.sol#546)

-lpTokenPayout = lpTokenPayout.mul(vestingPeriods).div(10) (REX_Rda.sol#549)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

RexDailyAuction._generateSupply(uint32) (REX_Rda.sol#652-680) uses a dangerous strict equality:

- remainderCheck == 0 (REX_Rda.sol#671)

RexDailyAuction._trackDonators(address,uint256) (REX_Rda.sol#473-479) uses a dangerous strict equality:

- donatorTotalBalance[_donatorAddress] == 0 (REX_Rda.sol#474)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in RexDailyAuction._fillLiquidityPool() (REX_Rda.sol#686-801):

External calls:

- REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#724)
- REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#725)
- BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount) (REX_Rda.sol#726)
- (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_busdAmount,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#728-742)

State variables written after the call(s):

- extraLiqBusdSent = extraLiqBusdSent.add(amountBUSD) (REX_Rda.sol#744)

Reentrancy in RexDailyAuction._fillLiquidityPool() (REX_Rda.sol#686-801):

External calls:

- REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#724)
- REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#725)
- BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount) (REX_Rda.sol#726)
- (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_busdAmount,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#728-742)
- REX_CONTRACT.mintSupply(address(this),_rexAmount_scope_3) (REX_Rda.sol#776)
- REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount_scope_3) (REX_Rda.sol#777)
- BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount_scope_2) (REX_Rda.sol#778)
- (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount_scope_3,_busdAmount_scope_2,0,0,address(0x0),block.timestamp.add(7200)) (REX_Rda.sol#780-794)

State variables written after the call(s):

- toSendToPairBusd = toSendToPairBusd.sub(amountBUSD_scope_5) (REX_Rda.sol#796)

Reentrancy in RexDailyAuction.sendLiquidityBUSD(uint256) (REX_Rda.sol#501-517):

External calls:

- require(bool,string)(BUSD_TOKEN.transferFrom(msg.sender,address(this),_busd_amount),REX: Transfer of BUSD failed.) (REX_Rda.sol#511)

State variables written after the call(s):

- INITIAL_LIQ_BUSD = INITIAL_LIQ_BUSD.add(_busd_amount) (REX_Rda.sol#514)
- liquidityBalances[msg.sender] = liquidityBalances[msg.sender].add(_busd_amount) (REX_Rda.sol#513)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

RexDailyAuction._fillLiquidityPool().amountBUSD_scope_5 (REX_Rda.sol#782) is a local variable never initialized

RexDailyAuction._fillLiquidityPool().reserveIn_scope_0 (REX_Rda.sol#757) is a local variable never initialized

RexDailyAuction._fillLiquidityPool().reserveOut_scope_1 (REX_Rda.sol#757) is a local variable never initialized

RexDailyAuction._createBPD(uint256).todaysNumOfBPD (REX_Rda.sol#991) is a local variable never initialized

RexDailyAuction._fillLiquidityPool().totalAdd (REX_Rda.sol#691) is a local variable never initialized

RexDailyAuction._fillLiquidityPool().amountREX_scope_4 (REX_Rda.sol#781) is a local variable never initialized

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

RexDailyAuction._fillLiquidityPool() (REX_Rda.sol#686-801) ignores return value by REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#725)
RexDailyAuction._fillLiquidityPool() (REX_Rda.sol#686-801) ignores return value by BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount) (REX_Rda.sol#726)
RexDailyAuction._fillLiquidityPool() (REX_Rda.sol#686-801) ignores return value by REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount_scope_3) (REX_Rda.sol#777)
RexDailyAuction._fillLiquidityPool() (REX_Rda.sol#686-801) ignores return value by BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount_scope_2) (REX_Rda.sol#778)
RexDailyAuction._sendInitialLiquidityToPCS() (REX_Rda.sol#809-853) ignores return value by REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#829)
RexDailyAuction._sendInitialLiquidityToPCS() (REX_Rda.sol#809-853) ignores return value by BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_startLiquidity) (REX_Rda.sol#830)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

Low & Informational:

Variable 'RexDailyAuction._fillLiquidityPool().amountBUSD (REX_Rda.sol#730)' in RexDailyAuction._fillLiquidityPool() (REX_Rda.sol#686-801) potentially used before declaration: (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount_scope_3,_busdAmount_scope_2,0,0,address(0x0),block.timestamp.add(7200)) (REX_Rda.sol#780-794)
Variable 'RexDailyAuction._fillLiquidityPool().amountREX (REX_Rda.sol#729)' in RexDailyAuction._fillLiquidityPool() (REX_Rda.sol#686-801) potentially used before declaration: (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount_scope_3,_busdAmount_scope_2,0,0,address(0x0),block.timestamp.add(7200)) (REX_Rda.sol#780-794)
Variable 'RexDailyAuction._fillLiquidityPool().reserveOut (REX_Rda.sol#703)' in RexDailyAuction._fillLiquidityPool() (REX_Rda.sol#686-801) potentially used before declaration: (reserveIn,reserveOut) = UNISWAP_PAIR.getReserves() (REX_Rda.sol#757)
Variable 'RexDailyAuction._fillLiquidityPool().reserveIn (REX_Rda.sol#703)' in RexDailyAuction._fillLiquidityPool() (REX_Rda.sol#686-801) potentially used before declaration: (reserveIn,reserveOut) = UNISWAP_PAIR.getReserves() (REX_Rda.sol#757)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Reentrancy in RexDailyAuction._fillLiquidityPool() (REX_Rda.sol#686-801):
  External calls:
  - REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#724)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#725)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount) (REX_Rda.sol#726)

- (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_busdAmount,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#728-742)

State variables written after the call(s):

- totalLpTokens = LP_TOKEN.balanceOf(address(this)) (REX_Rda.sol#746)

Reentrancy in RexDailyAuction._generateSupplyAndCheckPools(uint32) (REX_Rda.sol#607-647):

External calls:

- _sendInitialLiquidityToPCS() (REX_Rda.sol#616)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#828)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#829)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_startLiquidity) (REX_Rda.sol#830)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_startLiquidity,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#832-846)

State variables written after the call(s):

- _createBPD(400) (REX_Rda.sol#624)
  - BUSDPOOL = busdPoolTemp (REX_Rda.sol#1055)
- _setTreasuryRatio() (REX_Rda.sol#628)
  - BUSDPOOL = 0 (REX_Rda.sol#1216)
- _setTreasuryRatio() (REX_Rda.sol#628)
  - BUSDTREASURY = BUSD_TOKEN.balanceOf(address(this)) (REX_Rda.sol#1217)
- BUSDTREASURY = 0 (REX_Rda.sol#633)
- _createBPD(400) (REX_Rda.sol#624)
  - addressHitByRandom[who] = true (REX_Rda.sol#1022)
  - addressHitByRandom[who] = true (REX_Rda.sol#1049)
- _generateSupply(_donationDay) (REX_Rda.sol#620)
  - dailyGeneratedREX[_donationDay] = RX_DECIMALS.mul(DAY_ONE_SUPPLY) (REX_Rda.sol#657)
  - dailyGeneratedREX[_donationDay] = RX_DECIMALS.mul(DAY_TWO_SUPPLY) (REX_Rda.sol#658)
  - dailyGeneratedREX[_donationDay] = RX_DECIMALS.mul(DAY_THREE_SUPPLY.sub((uint256(_donationDay).sub(3)).mul(DAILY_DIFF_SUPPLY))) (REX_Rda.sol#660)
  - dailyGeneratedREX[_donationDay] = RX_DECIMALS.mul(DAY_LAST_SUPPLY) (REX_Rda.sol#661)
- _generateSupply(_donationDay) (REX_Rda.sol#620)
  - dailyRatio[_donationDay] = ratio (REX_Rda.sol#671)
  - dailyRatio[_donationDay] = ratio.add(1) (REX_Rda.sol#671)
- _generateSupply(_donationDay) (REX_Rda.sol#620)
  - g.totalGeneratedREX = g.totalGeneratedREX.add(dailyGeneratedREX[_donationDay]) (REX_Rda.sol#664)
  - g.generatedDays ++ (REX_Rda.sol#665)
- _createBPD(400) (REX_Rda.sol#624)
  - g.generatedBigPayDays ++ (REX_Rda.sol#1057)
- _createBPD(400) (REX_Rda.sol#624)
  - poolWasntEmpty = BUSDPOOL > 5000E18 (REX_Rda.sol#1056)

- _createBPD(400) (REX_Rda.sol#624)
  - randomBUSD[who] = randomBUSD[who].add(maxBUSDtoClaim) (REX_Rda.sol#1018)
  - randomBUSD[who] = randomBUSD[who].add(maxBUSDtoClaim) (REX_Rda.sol#1045)
- _createBPD(400) (REX_Rda.sol#624)
  - sumOfDonationsOfUnHit = sumOfDonationsOfUnHit.sub(originalDonation[who]) (REX_Rda.sol#1023)
  - sumOfDonationsOfUnHit = sumOfDonationsOfUnHit.sub(originalDonation[who]) (REX_Rda.sol#1050)
- _setTreasuryRatio() (REX_Rda.sol#628)
  - treasuryRatio = BUSDTREASURY.mul(1E10).div(sumOfDonationsOfUnHit) (REX_Rda.sol#1220)
  - treasuryRatio = 0 (REX_Rda.sol#1223)

Reentrancy in RexDailyAuction._generateSupplyAndCheckPools(uint32) (REX_Rda.sol#607-647):

External calls:

- _sendInitialLiquidityToPCS() (REX_Rda.sol#616)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#828)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#829)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_startLiquidity) (REX_Rda.sol#830)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_startLiquidity,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#832-846)
- BUSD_TOKEN.transfer(MARKETING_ADDR,amo) (REX_Rda.sol#634)
- sendValue(address(MARKETING_ADDR),address(this).balance) (REX_Rda.sol#635)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)

External calls sending eth:

- sendValue(address(MARKETING_ADDR),address(this).balance) (REX_Rda.sol#635)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)

State variables written after the call(s):

- lastCheckedSupplyDay = lastCheckedSupplyDay.add(1) (REX_Rda.sol#638)

Reentrancy in RexDailyAuction._reserveRex(address,address,uint256) (REX_Rda.sol#373-434):

External calls:

- BUSD_TOKEN.transfer(MARKETING_ADDR,_senderValue.div(20)) (REX_Rda.sol#410)
- BUSD_TOKEN.transfer(DEVELOPMENT_ADDR,_senderValue.div(20)) (REX_Rda.sol#411)

State variables written after the call(s):

- BUSDTREASURY = BUSDTREASURY.add(_senderValue.mul(uint256(5).sub(mrexRef)).div(100)) (REX_Rda.sol#426)
- BUSDTREASURY = BUSDTREASURY.add(_senderValue.mul(5).div(100)) (REX_Rda.sol#432)
- _addReferralToDay(_referralAddress,_currentRxDay(),amountREX) (REX_Rda.sol#419)
  - dailyTotalReferral[_donationDay] = dailyTotalReferral[_donationDay].add(_referralAmount) (REX_Rda.sol#465)
- referralBUSD[_referralAddress] = referralBUSD[_referralAddress].add(_senderValue.mul(mrexRef).div(100)) (REX_Rda.sol#425)
- _addReferralToDay(_referralAddress,_currentRxDay(),amountREX) (REX_Rda.sol#419)
  - referrerBalances[_referrer][_donationDay] = referrerBalances[_referrer][_donationDay].add(_referralAmount) (REX_Rda.sol#464)
- _trackReferrals(_referralAddress,amountREX) (REX_Rda.sol#420)
  - referrerTotalBalance[_referralAddress] = referrerTotalBalance[_referralAddress].add(_value) (REX_Rda.sol#491)

- _trackReferrals(_referralAddress,amountREX) (REX_Rda.sol#420)
  - uniqueReferrerCount ++ (REX_Rda.sol#489)
- _trackReferrals(_referralAddress,amountREX) (REX_Rda.sol#420)
  - uniqueReferrers[uniqueReferrerCount] = _referralAddress (REX_Rda.sol#488)

Reentrancy in RexDailyAuction._sendInitialLiquidityToPCS() (REX_Rda.sol#809-853):

External calls:

- REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#828)
- REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#829)
- BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_startLiquidity) (REX_Rda.sol#830)
- (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_startLiquidity,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#832-846)

State variables written after the call(s):

- totalLpTokens = LP_TOKEN.balanceOf(address(this)) (REX_Rda.sol#849)

Reentrancy in RexDailyAuction.claimBusdFromReferrals() (REX_Rda.sol#1159-1171):

External calls:

- supplyTrigger() (REX_Rda.sol#1160)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#828)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#829)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_startLiquidity) (REX_Rda.sol#830)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_startLiquidity,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#832-846)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#724)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#725)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount) (REX_Rda.sol#726)
  - BUSD_TOKEN.transfer(MARKETING_ADDR,amo) (REX_Rda.sol#634)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_busdAmount,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#728-742)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount_scope_3) (REX_Rda.sol#776)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount_scope_3) (REX_Rda.sol#777)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount_scope_2) (REX_Rda.sol#778)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount_scope_3,_busdAmount_scope_2,0,0,address(0x0),block.timestamp.add(7200)) (REX_Rda.sol#780-794)

External calls sending eth:

- supplyTrigger() (REX_Rda.sol#1160)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

State variables written after the call(s):

- referralBUSD[msg.sender] = 0 (REX_Rda.sol#1167)

Reentrancy in RexDailyAuction.claimRexFromDonations() (REX_Rda.sol#1102-1127):

External calls:

- supplyTrigger() (REX_Rda.sol#1103)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#828)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#829)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_startLiquidity) (REX_Rda.sol#830)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_startLiquidity,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#832-846)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#724)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#725)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount) (REX_Rda.sol#726)
  - BUSD_TOKEN.transfer(MARKETING_ADDR,amo) (REX_Rda.sol#634)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_busdAmount,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#728-742)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount_scope_3) (REX_Rda.sol#776)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount_scope_3) (REX_Rda.sol#777)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount_scope_2) (REX_Rda.sol#778)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount_scope_3,_busdAmount_scope_2,0,0,address(0x0),block.timestamp.add(7200)) (REX_Rda.sol#780-794)

External calls sending eth:

- supplyTrigger() (REX_Rda.sol#1103)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

State variables written after the call(s):

- _removeEligibleAddr(msg.sender) (REX_Rda.sol#1121)
  - addressBPDExcluded[_userAddress] = true (REX_Rda.sol#922)
- donatorBalancesDrawn[msg.sender][i] = true (REX_Rda.sol#1115)
- donatorTotalRexReceived[msg.sender] = donatorTotalRexReceived[msg.sender].add(_payout) (REX_Rda.sol#1123)
- _removeEligibleAddr(msg.sender) (REX_Rda.sol#1121)
  - userIndicesBPD[addressToMove] = indexToDelete (REX_Rda.sol#929)
  - userIndicesBPD[_userAddress] = 0 (REX_Rda.sol#930)

Reentrancy in RexDailyAuction.claimRexFromReferrals() (REX_Rda.sol#1134-1154):

External calls:

- supplyTrigger() (REX_Rda.sol#1135)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#828)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#829)

- BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_startLiquidity) (REX_Rda.sol#830)

- (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_startLiquidity,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#832-846)

- REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#724)

- REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#725)

- BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount) (REX_Rda.sol#726)

- BUSD_TOKEN.transfer(MARKETING_ADDR,amo) (REX_Rda.sol#634)

- (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_busdAmount,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#728-742)

- REX_CONTRACT.mintSupply(address(this),_rexAmount_scope_3) (REX_Rda.sol#776)

- REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount_scope_3) (REX_Rda.sol#777)

- BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount_scope_2) (REX_Rda.sol#778)

- (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount_scope_3,_busdAmount_scope_2,0,0,address(0x0),block.timestamp.add(7200)) (REX_Rda.sol#780-794)

External calls sending eth:

- supplyTrigger() (REX_Rda.sol#1135)

- (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)

- msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

State variables written after the call(s):

- referrerBalancesDrawn[msg.sender][i] = true (REX_Rda.sol#1145)

Reentrancy in RexDailyAuction.claimStakeFromDonations(uint32) (REX_Rda.sol#1066-1094):

External calls:

- supplyTrigger() (REX_Rda.sol#1069)

- (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)

- REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#828)

- REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#829)

- BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_startLiquidity) (REX_Rda.sol#830)

- (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_startLiquidity,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#832-846)

- REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#724)

- REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#725)

- BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount) (REX_Rda.sol#726)

- BUSD_TOKEN.transfer(MARKETING_ADDR,amo) (REX_Rda.sol#634)

- (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_busdAmount,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#728-742)

- REX_CONTRACT.mintSupply(address(this),_rexAmount_scope_3) (REX_Rda.sol#776)

- REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount_scope_3) (REX_Rda.sol#777)

- BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount_scope_2) (REX_Rda.sol#778)
- (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount_scope_3,_busdAmount_scope_2,0,0,address(0x0),block.timestamp.add(7200)) (REX_Rda.sol#780-794)

External calls sending eth:

- supplyTrigger() (REX_Rda.sol#1069)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

State variables written after the call(s):

- donatorBalancesDrawn[msg.sender][i] = true (REX_Rda.sol#1082)
- donatorTotalRexReceived[msg.sender] = donatorTotalRexReceived[msg.sender].add(_payout) (REX_Rda.sol#1090)
- _addEligibleAddr(msg.sender) (REX_Rda.sol#1088)
  - userIndicesBPD[_userAddress] = userAddressesBPD.length - 1 (REX_Rda.sol#901)
  - userIndicesBPD[userAddressesBPD[_pos]] = userAddressesBPD.length - 1 (REX_Rda.sol#907)
  - userIndicesBPD[_userAddress] = _pos (REX_Rda.sol#909)

Reentrancy in RexDailyAuction.donateBUSD(uint256,address) (REX_Rda.sol#352-366):

External calls:

- require(bool,string)(BUSD_TOKEN.transferFrom(msg.sender,address(this),_busd_amount),REX: Transfer of BUSD failed.) (REX_Rda.sol#363)
- _reserveRex(_referralAddress,msg.sender,_busd_amount) (REX_Rda.sol#365)
  - BUSD_TOKEN.transfer(MARKETING_ADDR,_senderValue.div(20)) (REX_Rda.sol#410)
  - BUSD_TOKEN.transfer(DEVELOPMENT_ADDR,_senderValue.div(20)) (REX_Rda.sol#411)
- supplyTrigger() (REX_Rda.sol#353)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#828)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#829)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_startLiquidity) (REX_Rda.sol#830)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_startLiquidity,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#832-846)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#724)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#725)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount) (REX_Rda.sol#726)
  - BUSD_TOKEN.transfer(MARKETING_ADDR,amo) (REX_Rda.sol#634)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_busdAmount,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#728-742)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount_scope_3) (REX_Rda.sol#776)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount_scope_3) (REX_Rda.sol#777)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount_scope_2) (REX_Rda.sol#778)

- (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount_scope_3,_busdAmount_scope_2,0,0,address(0x0),block.timestamp.add(7200)) (REX_Rda.sol#780-794)

External calls sending eth:

- supplyTrigger() (REX_Rda.sol#353)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

State variables written after the call(s):

- _reserveRex(_referralAddress,msg.sender,_busd_amount) (REX_Rda.sol#365)
  - uniqueDonators[uniqueDonatorCount] = _donatorAddress (REX_Rda.sol#475)
- _reserveRex(_referralAddress,msg.sender,_busd_amount) (REX_Rda.sol#365)
  - uniqueReferrers[uniqueReferrerCount] = _referralAddress (REX_Rda.sol#488)

Reentrancy in RexDailyAuction.supplyTrigger() (REX_Rda.sol#298-303):

External calls:

- _dailyDistributionRoutine() (REX_Rda.sol#300)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#828)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#829)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_startLiquidity) (REX_Rda.sol#830)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_startLiquidity,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#832-846)
  - BUSD_TOKEN.transfer(MARKETING_ADDR,amo) (REX_Rda.sol#634)
- _fillLiquidityPool() (REX_Rda.sol#302)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#724)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#725)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount) (REX_Rda.sol#726)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_busdAmount,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#728-742)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount_scope_3) (REX_Rda.sol#776)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount_scope_3) (REX_Rda.sol#777)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount_scope_2) (REX_Rda.sol#778)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount_scope_3,_busdAmount_scope_2,0,0,address(0x0),block.timestamp.add(7200)) (REX_Rda.sol#780-794)

External calls sending eth:

- _dailyDistributionRoutine() (REX_Rda.sol#300)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

State variables written after the call(s):

- _fillLiquidityPool() (REX_Rda.sol#302)

- totalLpTokens = LP_TOKEN.balanceOf(address(this)) (REX_Rda.sol#746)

Reentrancy in RexDailyAuction.withdrawLPTokens() (REX_Rda.sol#521-534):

External calls:

- supplyTrigger() (REX_Rda.sol#522)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#828)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#829)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_startLiquidity) (REX_Rda.sol#830)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_startLiquidity,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#832-846)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#724)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#725)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount) (REX_Rda.sol#726)
  - BUSD_TOKEN.transfer(MARKETING_ADDR,amo) (REX_Rda.sol#634)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_busdAmount,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#728-742)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount_scope_3) (REX_Rda.sol#776)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount_scope_3) (REX_Rda.sol#777)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount_scope_2) (REX_Rda.sol#778)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount_scope_3,_busdAmount_scope_2,0,0,address(0x0),block.timestamp.add(7200)) (REX_Rda.sol#780-794)

External calls sending eth:

- supplyTrigger() (REX_Rda.sol#522)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

State variables written after the call(s):

- liquidityBalancesDrawn[msg.sender] = liquidityBalancesDrawn[msg.sender].add(lpTokensPayout) (REX_Rda.sol#530)

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

Reentrancy in RexDailyAuction._fillLiquidityPool() (REX_Rda.sol#686-801):

External calls:

- REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#724)
- REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#725)
- BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount) (REX_Rda.sol#726)
- (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_busdAmount,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#728-742)

Event emitted after the call(s):

- LiquidityGenerated(_currentRxDay(),amountBUSD,amountREX) (REX_Rda.sol#748)

Reentrancy in RexDailyAuction._fillLiquidityPool() (REX_Rda.sol#686-801):

External calls:

- REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#724)
- REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#725)
- BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount) (REX_Rda.sol#726)
- (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_busdAmount,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#728-742)
- REX_CONTRACT.mintSupply(address(this),_rexAmount_scope_3) (REX_Rda.sol#776)
- REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount_scope_3) (REX_Rda.sol#777)
- BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount_scope_2) (REX_Rda.sol#778)
- (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount_scope_3,_busdAmount_scope_2,0,0,address(0x0),block.timestamp.add(7200)) (REX_Rda.sol#780-794)

Event emitted after the call(s):

- LiquidityGenerated(_currentRxDay(),amountBUSD_scope_5,amountREX_scope_4) (REX_Rda.sol#798)

Reentrancy in RexDailyAuction._generateSupplyAndCheckPools(uint32) (REX_Rda.sol#607-647):

External calls:

- _sendInitialLiquidityToPCS() (REX_Rda.sol#616)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#828)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#829)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_startLiquidity) (REX_Rda.sol#830)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_startLiquidity,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#832-846)

Event emitted after the call(s):

- DistributedBigPayDay(g.generatedBigPayDays,maxNumOfBPD,todaysNumOfBPD,busdPoolStart,BUSDPOOL) (REX_Rda.sol#1058)
  - _createBPD(400) (REX_Rda.sol#624)
- SupplyGenerated(_donationDay,dailyGeneratedREX[_donationDay]) (REX_Rda.sol#673)
  - _generateSupply(_donationDay) (REX_Rda.sol#620)
- SupplyGenerated(_donationDay,uint256(0)) (REX_Rda.sol#677)
  - _generateSupply(_donationDay) (REX_Rda.sol#620)
- TreasuryGenerated(BUSDTREASURY,treasuryRatio) (REX_Rda.sol#1225)
  - _setTreasuryRatio() (REX_Rda.sol#628)

Reentrancy in RexDailyAuction._generateSupplyAndCheckPools(uint32) (REX_Rda.sol#607-647):

External calls:

- _sendInitialLiquidityToPCS() (REX_Rda.sol#616)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#828)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#829)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_startLiquidity) (REX_Rda.sol#830)

- (amountREX,amountBUSD) =
UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_startLiquidity,0,0,address(this),bl
ock.timestamp.add(7200)) (REX_Rda.sol#832-846)
- BUSD_TOKEN.transfer(MARKETING_ADDR,amo) (REX_Rda.sol#634)
- sendValue(address(MARKETING_ADDR),address(this).balance) (REX_Rda.sol#635)
- (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)

External calls sending eth:
- sendValue(address(MARKETING_ADDR),address(this).balance) (REX_Rda.sol#635)
- (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
- msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

Event emitted after the call(s):
- GasRefunded(msg.sender,refundBNB) (REX_Rda.sol#644)

Reentrancy in RexDailyAuction._reserveRex(address,address,uint256) (REX_Rda.sol#373-434):

External calls:
- BUSD_TOKEN.transfer(MARKETING_ADDR,_senderValue.div(20)) (REX_Rda.sol#410)
- BUSD_TOKEN.transfer(DEVELOPMENT_ADDR,_senderValue.div(20)) (REX_Rda.sol#411)

Event emitted after the call(s):
- ReferralAdded(_referralAddress,_senderAddress,amountREX) (REX_Rda.sol#428)

Reentrancy in RexDailyAuction._sendInitialLiquidityToPCS() (REX_Rda.sol#809-853):

External calls:
- REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#828)
- REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#829)
- BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_startLiquidity) (REX_Rda.sol#830)
- (amountREX,amountBUSD) =
UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_startLiquidity,0,0,address(this),bl
ock.timestamp.add(7200)) (REX_Rda.sol#832-846)

Event emitted after the call(s):
- LiquidityGenerated(0,amountBUSD,amountREX) (REX_Rda.sol#851)

Reentrancy in RexDailyAuction.claimBusdFromBPD() (REX_Rda.sol#1176-1188):

External calls:
- BUSD_TOKEN.transfer(msg.sender,claimed) (REX_Rda.sol#1186)
- supplyTrigger() (REX_Rda.sol#1177)
- (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
- REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#828)
- REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#829)
- BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_startLiquidity) (REX_Rda.sol#830)
- (amountREX,amountBUSD) =
UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_startLiquidity,0,0,address(this),bl
ock.timestamp.add(7200)) (REX_Rda.sol#832-846)
- REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#724)
- REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#725)
- BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount) (REX_Rda.sol#726)

- BUSD_TOKEN.transfer(MARKETING_ADDR,amo) (REX_Rda.sol#634)
- (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_busdAmount,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#728-742)
- REX_CONTRACT.mintSupply(address(this),_rexAmount_scope_3) (REX_Rda.sol#776)
- REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount_scope_3) (REX_Rda.sol#777)
- BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount_scope_2) (REX_Rda.sol#778)
- (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount_scope_3,_busdAmount_scope_2,0,0,address(0x0),block.timestamp.add(7200)) (REX_Rda.sol#780-794)

External calls sending eth:

- supplyTrigger() (REX_Rda.sol#1177)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

Event emitted after the call(s):

- ClaimedBusdFromBPD(msg.sender,claimed) (REX_Rda.sol#1187)

Reentrancy in RexDailyAuction.claimBusdFromReferrals() (REX_Rda.sol#1159-1171):

External calls:

- BUSD_TOKEN.transfer(msg.sender,claimed) (REX_Rda.sol#1169)
- supplyTrigger() (REX_Rda.sol#1160)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#828)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#829)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_startLiquidity) (REX_Rda.sol#830)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_startLiquidity,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#832-846)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#724)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#725)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount) (REX_Rda.sol#726)
  - BUSD_TOKEN.transfer(MARKETING_ADDR,amo) (REX_Rda.sol#634)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_busdAmount,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#728-742)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount_scope_3) (REX_Rda.sol#776)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount_scope_3) (REX_Rda.sol#777)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount_scope_2) (REX_Rda.sol#778)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount_scope_3,_busdAmount_scope_2,0,0,address(0x0),block.timestamp.add(7200)) (REX_Rda.sol#780-794)

External calls sending eth:

- supplyTrigger() (REX_Rda.sol#1160)

- (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
- msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

Event emitted after the call(s):
- ClaimedBusdFromReferrals(msg.sender,claimed) (REX_Rda.sol#1170)

Reentrancy in RexDailyAuction.claimRexFromDonations() (REX_Rda.sol#1102-1127):

External calls:
- REX_CONTRACT.mintSupply(msg.sender,_payout) (REX_Rda.sol#1124)
- supplyTrigger() (REX_Rda.sol#1103)
- (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
- REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#828)
- REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#829)
- BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_startLiquidity) (REX_Rda.sol#830)
- (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_startLiquidity,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#832-846)
- REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#724)
- REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#725)
- BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount) (REX_Rda.sol#726)
- BUSD_TOKEN.transfer(MARKETING_ADDR,amo) (REX_Rda.sol#634)
- (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_busdAmount,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#728-742)
- REX_CONTRACT.mintSupply(address(this),_rexAmount_scope_3) (REX_Rda.sol#776)
- REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount_scope_3) (REX_Rda.sol#777)
- BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount_scope_2) (REX_Rda.sol#778)
- (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount_scope_3,_busdAmount_scope_2,0,0,address(0x0),block.timestamp.add(7200)) (REX_Rda.sol#780-794)

External calls sending eth:
- supplyTrigger() (REX_Rda.sol#1103)
- (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
- msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

Event emitted after the call(s):
- ClaimedRexFromAuctions(msg.sender,_payout) (REX_Rda.sol#1125)

Reentrancy in RexDailyAuction.claimRexFromReferrals() (REX_Rda.sol#1134-1154):

External calls:
- REX_CONTRACT.mintSupply(msg.sender,_payout) (REX_Rda.sol#1151)
- supplyTrigger() (REX_Rda.sol#1135)
- (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
- REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#828)
- REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#829)
- BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_startLiquidity) (REX_Rda.sol#830)

- (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_startLiquidity,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#832-846)
- REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#724)
- REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#725)
- BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount) (REX_Rda.sol#726)
- BUSD_TOKEN.transfer(MARKETING_ADDR,amo) (REX_Rda.sol#634)
- (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_busdAmount,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#728-742)
- REX_CONTRACT.mintSupply(address(this),_rexAmount_scope_3) (REX_Rda.sol#776)
- REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount_scope_3) (REX_Rda.sol#777)
- BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount_scope_2) (REX_Rda.sol#778)
- (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount_scope_3,_busdAmount_scope_2,0,0,address(0x0),block.timestamp.add(7200)) (REX_Rda.sol#780-794)

External calls sending eth:

- supplyTrigger() (REX_Rda.sol#1135)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

Event emitted after the call(s):

- ClaimedRexFromReferrals(msg.sender,_payout) (REX_Rda.sol#1152)

Reentrancy in RexDailyAuction.claimStakeFromDonations(uint32) (REX_Rda.sol#1066-1094):

External calls:

- REX_CONTRACT.createStake(msg.sender,_payout,_stakingDays,0,true) (REX_Rda.sol#1091)
- supplyTrigger() (REX_Rda.sol#1069)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#828)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#829)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_startLiquidity) (REX_Rda.sol#830)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_startLiquidity,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#832-846)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#724)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#725)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount) (REX_Rda.sol#726)
  - BUSD_TOKEN.transfer(MARKETING_ADDR,amo) (REX_Rda.sol#634)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_busdAmount,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#728-742)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount_scope_3) (REX_Rda.sol#776)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount_scope_3) (REX_Rda.sol#777)

- BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount_scope_2) (REX_Rda.sol#778)
- (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount_scope_3,_busdAmount_scope_2,0,0,address(0x0),block.timestamp.add(7200)) (REX_Rda.sol#780-794)

External calls sending eth:

- supplyTrigger() (REX_Rda.sol#1069)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

Event emitted after the call(s):

- ClaimedStakeFromAuctions(msg.sender,_payout) (REX_Rda.sol#1092)

Reentrancy in RexDailyAuction.donateBUSD(uint256,address) (REX_Rda.sol#352-366):

External calls:

- require(bool,string)(BUSD_TOKEN.transferFrom(msg.sender,address(this),_busd_amount),REX: Transfer of BUSD failed.) (REX_Rda.sol#363)
- _reserveRex(_referralAddress,msg.sender,_busd_amount) (REX_Rda.sol#365)
  - BUSD_TOKEN.transfer(MARKETING_ADDR,_senderValue.div(20)) (REX_Rda.sol#410)
  - BUSD_TOKEN.transfer(DEVELOPMENT_ADDR,_senderValue.div(20)) (REX_Rda.sol#411)
- supplyTrigger() (REX_Rda.sol#353)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#828)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#829)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_startLiquidity) (REX_Rda.sol#830)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_startLiquidity,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#832-846)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#724)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#725)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount) (REX_Rda.sol#726)
  - BUSD_TOKEN.transfer(MARKETING_ADDR,amo) (REX_Rda.sol#634)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_busdAmount,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#728-742)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount_scope_3) (REX_Rda.sol#776)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount_scope_3) (REX_Rda.sol#777)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount_scope_2) (REX_Rda.sol#778)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount_scope_3,_busdAmount_scope_2,0,0,address(0x0),block.timestamp.add(7200)) (REX_Rda.sol#780-794)

External calls sending eth:

- supplyTrigger() (REX_Rda.sol#353)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

Event emitted after the call(s):

- DonationReceived(_senderAddress,_donationDay,_donationBalance) (REX_Rda.sol#454)
  - _reserveRex(_referralAddress,msg.sender,_busd_amount) (REX_Rda.sol#365)
- ReferralAdded(_referralAddress,_senderAddress,amountREX) (REX_Rda.sol#428)
  - _reserveRex(_referralAddress,msg.sender,_busd_amount) (REX_Rda.sol#365)

Reentrancy in RexDailyAuction.sendLiquidityBUSD(uint256) (REX_Rda.sol#501-517):

External calls:

- require(bool,string)(BUSD_TOKEN.transferFrom(msg.sender,address(this),_busd_amount),REX: Transfer of BUSD failed.) (REX_Rda.sol#511)

Event emitted after the call(s):

- LiquidityReceived(msg.sender,_busd_amount) (REX_Rda.sol#516)

Reentrancy in RexDailyAuction.supplyTrigger() (REX_Rda.sol#298-303):

External calls:

- _dailyDistributionRoutine() (REX_Rda.sol#300)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#828)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#829)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_startLiquidity) (REX_Rda.sol#830)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_startLiquidity,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#832-846)
  - BUSD_TOKEN.transfer(MARKETING_ADDR,amo) (REX_Rda.sol#634)
- _fillLiquidityPool() (REX_Rda.sol#302)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#724)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#725)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount) (REX_Rda.sol#726)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_busdAmount,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#728-742)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount_scope_3) (REX_Rda.sol#776)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount_scope_3) (REX_Rda.sol#777)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount_scope_2) (REX_Rda.sol#778)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount_scope_3,_busdAmount_scope_2,0,0,address(0x0),block.timestamp.add(7200)) (REX_Rda.sol#780-794)

External calls sending eth:

- _dailyDistributionRoutine() (REX_Rda.sol#300)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

Event emitted after the call(s):

- LiquidityGenerated(_currentRxDay(),amountBUSD,amountREX) (REX_Rda.sol#748)
  - _fillLiquidityPool() (REX_Rda.sol#302)

- LiquidityGenerated(_currentRxDay(),amountBUSD_scope_5,amountREX_scope_4) (REX_Rda.sol#798)
  - _fillLiquidityPool() (REX_Rda.sol#302)

Reentrancy in RexDailyAuction.withdrawLPTokens() (REX_Rda.sol#521-534):

External calls:

- LP_TOKEN.transfer(msg.sender,lpTokensPayout) (REX_Rda.sol#531)
- supplyTrigger() (REX_Rda.sol#522)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#828)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#829)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_startLiquidity) (REX_Rda.sol#830)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_startLiquidity,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#832-846)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount) (REX_Rda.sol#724)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount) (REX_Rda.sol#725)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount) (REX_Rda.sol#726)
  - BUSD_TOKEN.transfer(MARKETING_ADDR,amo) (REX_Rda.sol#634)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount,_busdAmount,0,0,address(this),block.timestamp.add(7200)) (REX_Rda.sol#728-742)
  - REX_CONTRACT.mintSupply(address(this),_rexAmount_scope_3) (REX_Rda.sol#776)
  - REX_CONTRACT.approve(address(UNISWAP_ROUTER),_rexAmount_scope_3) (REX_Rda.sol#777)
  - BUSD_TOKEN.approve(address(UNISWAP_ROUTER),_busdAmount_scope_2) (REX_Rda.sol#778)
  - (amountREX,amountBUSD) = UNISWAP_ROUTER.addLiquidity(address(REX_CONTRACT),busd_address,_rexAmount_scope_3,_busdAmount_scope_2,0,0,address(0x0),block.timestamp.add(7200)) (REX_Rda.sol#780-794)

External calls sending eth:

- supplyTrigger() (REX_Rda.sol#522)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

Event emitted after the call(s):

- LPtokensWithdrawn(msg.sender,lpTokensPayout) (REX_Rda.sol#533)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

RexDailyAuction._fillLiquidityPool() (REX_Rda.sol#686-801) uses timestamp for comparisons

Dangerous comparisons:

- (_currentRxDay() == 203 && extraLiqBusdSent < uint256(200).mul(EXTRA_DAILY_LIQ)) || (_currentRxDay() < 203 && extraLiqBusdSent < uint256(_currentRxDay().sub(2)).mul(EXTRA_DAILY_LIQ)) (REX_Rda.sol#699-700)
- toSendToPairBusd > uint256(1000E18).sub(totalAdd) (REX_Rda.sol#759-761)

RexDailyAuction._shuffleBPDlist() (REX_Rda.sol#937-955) uses timestamp for comparisons

Dangerous comparisons:

- _posA != _posB (REX_Rda.sol#946)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

RexDailyAuction._notContract(address) (REX_Rda.sol#1308-1309) uses assembly
- INLINE ASM (REX_Rda.sol#1309)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

RexSafeMath32.div(uint32,uint32) (REX_Rda.sol#1398-1402) is never used and should be removed
RexSafeMath32.mul(uint32,uint32) (REX_Rda.sol#1386-1396) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.7.4 (REX_Rda.sol#3) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in RexDailyAuction.sendValue(address,uint256) (REX_Rda.sol#1311-1312):
- (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function IREXToken.UNISWAP_PAIR() (REX_Rda.sol#138-140) is not in mixedCase
Parameter RexDailyAuction.initContracts(address,address)._rex (REX_Rda.sol#329) is not in mixedCase
Parameter RexDailyAuction.initContracts(address,address)._trex (REX_Rda.sol#329) is not in mixedCase
Parameter RexDailyAuction.donateBUSD(uint256,address)._busd_amount (REX_Rda.sol#352) is not in mixedCase
Parameter RexDailyAuction.donateBUSD(uint256,address)._referralAddress (REX_Rda.sol#352) is not in mixedCase
Parameter RexDailyAuction.sendLiquidityBUSD(uint256)._busd_amount (REX_Rda.sol#501) is not in mixedCase
Parameter RexDailyAuction.isBPDeligibleAddr(address)._userAddress (REX_Rda.sol#878) is not in mixedCase
Function RexDailyAuction._createUserBPD() (REX_Rda.sol#968-975) is not in mixedCase
Parameter RexDailyAuction.claimStakeFromDonations(uint32)._stakingDays (REX_Rda.sol#1067) is not in mixedCase
Parameter RexDailyAuction.auctionSupplyOnDay(uint32)._donationDay (REX_Rda.sol#1284) is not in mixedCase
Parameter RexDailyAuction.auctionStatsOfDay(uint32)._donationDay (REX_Rda.sol#1293) is not in mixedCase
Function RexDailyAuction._currentRxDay() (REX_Rda.sol#1304-1306) is not in mixedCase
Variable RexDailyAuction.TOKEN_DEFINER (REX_Rda.sol#192) is not in mixedCase
Variable RexDailyAuction.GAS_REFUNDER (REX_Rda.sol#193) is not in mixedCase
Variable RexDailyAuction.UNISWAP_PAIR (REX_Rda.sol#194) is not in mixedCase
Variable RexDailyAuction.REX_CONTRACT (REX_Rda.sol#195) is not in mixedCase
Variable RexDailyAuction.TREX_TOKEN (REX_Rda.sol#196) is not in mixedCase
Variable RexDailyAuction.MREX_TOKEN (REX_Rda.sol#197) is not in mixedCase
Variable RexDailyAuction.BUSD_TOKEN (REX_Rda.sol#198) is not in mixedCase
Variable RexDailyAuction.LP_TOKEN (REX_Rda.sol#199) is not in mixedCase
Constant RexDailyAuction.mrex_address (REX_Rda.sol#202) is not in UPPER_CASE_WITH_UNDERSCORES
Constant RexDailyAuction.busd_address (REX_Rda.sol#203) is not in UPPER_CASE_WITH_UNDERSCORES
Variable RexDailyAuction.INITIAL_LIQ_BUSD (REX_Rda.sol#269) is not in mixedCase
Variable RexDailyAuction.EXTRA_DAILY_LIQ (REX_Rda.sol#270) is not in mixedCase
Variable RexDailyAuction.BUSDPOOL (REX_Rda.sol#275) is not in mixedCase

Variable RexDailyAuction.BUSDTREASURY (REX_Rda.sol#276) is not in mixedCase

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Reentrancy in RexDailyAuction._generateSupplyAndCheckPools(uint32) (REX_Rda.sol#607-647):

External calls:

- msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

External calls sending eth:

- sendValue(address(MARKETING_ADDR),address(this).balance) (REX_Rda.sol#635)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
- msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

Event emitted after the call(s):

- GasRefunded(msg.sender,refundBNB) (REX_Rda.sol#644)

Reentrancy in RexDailyAuction.claimBusdFromBPD() (REX_Rda.sol#1176-1188):

External calls:

- supplyTrigger() (REX_Rda.sol#1177)
  - msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

External calls sending eth:

- supplyTrigger() (REX_Rda.sol#1177)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

State variables written after the call(s):

- g.totalClaimedRandomBUSD = g.totalClaimedRandomBUSD.add(claimed) (REX_Rda.sol#1185)
- randomBUSD[msg.sender] = 0 (REX_Rda.sol#1184)

Event emitted after the call(s):

- ClaimedBusdFromBPD(msg.sender,claimed) (REX_Rda.sol#1187)

Reentrancy in RexDailyAuction.claimBusdFromReferrals() (REX_Rda.sol#1159-1171):

External calls:

- supplyTrigger() (REX_Rda.sol#1160)
  - msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

External calls sending eth:

- supplyTrigger() (REX_Rda.sol#1160)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

State variables written after the call(s):

- g.totalClaimedReferralBUSD = g.totalClaimedReferralBUSD.add(claimed) (REX_Rda.sol#1168)
- referralBUSD[msg.sender] = 0 (REX_Rda.sol#1167)

Event emitted after the call(s):

- ClaimedBusdFromReferrals(msg.sender,claimed) (REX_Rda.sol#1170)

Reentrancy in RexDailyAuction.claimBusdFromTREASURY() (REX_Rda.sol#1193-1208):

External calls:

- supplyTrigger() (REX_Rda.sol#1194)
  - msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

External calls sending eth:

- supplyTrigger() (REX_Rda.sol#1194)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

State variables written after the call(s):

- BUSDTREASURY = BUSDTREASURY.sub(claimed) (REX_Rda.sol#1206)
- addressHitByRandom[msg.sender] = true (REX_Rda.sol#1205)

Reentrancy in RexDailyAuction.claimRexFromDonations() (REX_Rda.sol#1102-1127):

External calls:

- supplyTrigger() (REX_Rda.sol#1103)
  - msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

External calls sending eth:

- supplyTrigger() (REX_Rda.sol#1103)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

State variables written after the call(s):

- _removeEligibleAddr(msg.sender) (REX_Rda.sol#1121)
  - addressBPDExcluded[_userAddress] = true (REX_Rda.sol#922)
- donatorBalancesDrawn[msg.sender][i] = true (REX_Rda.sol#1115)
- donatorTotalRexReceived[msg.sender] = donatorTotalRexReceived[msg.sender].add(_payout) (REX_Rda.sol#1123)
- g.totalClaimedDonationREX = g.totalClaimedDonationREX.add(_payout) (REX_Rda.sol#1122)
- _removeEligibleAddr(msg.sender) (REX_Rda.sol#1121)
  - userAddressesBPD[indexToDelete] = addressToMove (REX_Rda.sol#928)
  - userAddressesBPD.pop() (REX_Rda.sol#931)
- _removeEligibleAddr(msg.sender) (REX_Rda.sol#1121)
  - userIndicesBPD[addressToMove] = indexToDelete (REX_Rda.sol#929)
  - userIndicesBPD[_userAddress] = 0 (REX_Rda.sol#930)

Event emitted after the call(s):

- ClaimedRexFromAuctions(msg.sender,_payout) (REX_Rda.sol#1125)

Reentrancy in RexDailyAuction.claimRexFromReferrals() (REX_Rda.sol#1134-1154):

External calls:

- supplyTrigger() (REX_Rda.sol#1135)
  - msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

External calls sending eth:

- supplyTrigger() (REX_Rda.sol#1135)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

State variables written after the call(s):

- g.totalClaimedReferralREX = g.totalClaimedReferralREX.add(_payout) (REX_Rda.sol#1150)
- referrerBalancesDrawn[msg.sender][i] = true (REX_Rda.sol#1145)

Event emitted after the call(s):

- ClaimedRexFromReferrals(msg.sender,_payout) (REX_Rda.sol#1152)

Reentrancy in RexDailyAuction.claimStakeFromDonations(uint32) (REX_Rda.sol#1066-1094):

External calls:

- supplyTrigger() (REX_Rda.sol#1069)
  - msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

External calls sending eth:

- supplyTrigger() (REX_Rda.sol#1069)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

State variables written after the call(s):

- donatorBalancesDrawn[msg.sender][i] = true (REX_Rda.sol#1082)
- donatorTotalRexReceived[msg.sender] = donatorTotalRexReceived[msg.sender].add(_payout) (REX_Rda.sol#1090)
- g.totalClaimedDonationREX = g.totalClaimedDonationREX.add(_payout) (REX_Rda.sol#1089)
- _addEligibleAddr(msg.sender) (REX_Rda.sol#1088)
  - userAddressesBPD.push(_userAddress) (REX_Rda.sol#900)
  - userAddressesBPD.push(userAddressesBPD[_pos]) (REX_Rda.sol#906)
  - userAddressesBPD[_pos] = _userAddress (REX_Rda.sol#908)
- _addEligibleAddr(msg.sender) (REX_Rda.sol#1088)
  - userIndicesBPD[_userAddress] = userAddressesBPD.length - 1 (REX_Rda.sol#901)
  - userIndicesBPD[userAddressesBPD[_pos]] = userAddressesBPD.length - 1 (REX_Rda.sol#907)
  - userIndicesBPD[_userAddress] = _pos (REX_Rda.sol#909)

Event emitted after the call(s):

- ClaimedStakeFromAuctions(msg.sender,_payout) (REX_Rda.sol#1092)

Reentrancy in RexDailyAuction.donateBUSD(uint256,address) (REX_Rda.sol#352-366):

External calls:

- supplyTrigger() (REX_Rda.sol#353)
  - msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

External calls sending eth:

- supplyTrigger() (REX_Rda.sol#353)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

State variables written after the call(s):

- _reserveRex(_referralAddress,msg.sender,_busd_amount) (REX_Rda.sol#365)
  - BUSDPOOL = BUSDPOOL.add(_senderValue.mul(75).div(100)) (REX_Rda.sol#408)
- _reserveRex(_referralAddress,msg.sender,_busd_amount) (REX_Rda.sol#365)
  - BUSDTREASURY = BUSDTREASURY.add(_senderValue.mul(uint256(5).sub(mrexRef)).div(100))
(REX_Rda.sol#426)
  - BUSDTREASURY = BUSDTREASURY.add(_senderValue.mul(5).div(100)) (REX_Rda.sol#432)
- _reserveRex(_referralAddress,msg.sender,_busd_amount) (REX_Rda.sol#365)
  - dailyTotalDonation[_donationDay] = dailyTotalDonation[_donationDay].add(_donationBalance) (REX_Rda.sol#452)
- _reserveRex(_referralAddress,msg.sender,_busd_amount) (REX_Rda.sol#365)
  - dailyTotalReferral[_donationDay] = dailyTotalReferral[_donationDay].add(_referralAmount) (REX_Rda.sol#465)
- _reserveRex(_referralAddress,msg.sender,_busd_amount) (REX_Rda.sol#365)

- donatorAccountCount[_donationDay] ++ (REX_Rda.sol#449)

- _reserveRex(_referralAddress,msg.sender,_busd_amount) (REX_Rda.sol#365)

- donatorBalances[_senderAddress][_donationDay] = donatorBalances[_senderAddress][_donationDay].add(_donationBalance) (REX_Rda.sol#451)

- _reserveRex(_referralAddress,msg.sender,_busd_amount) (REX_Rda.sol#365)

- donatorTotalBalance[_donatorAddress] = donatorTotalBalance[_donatorAddress].add(_value) (REX_Rda.sol#478)

- _reserveRex(_referralAddress,msg.sender,_busd_amount) (REX_Rda.sol#365)

- g.totalDonatedBUSD = g.totalDonatedBUSD.add(_senderValue) (REX_Rda.sol#406)

- _reserveRex(_referralAddress,msg.sender,_busd_amount) (REX_Rda.sol#365)

- originalDonation[_senderAddress] = originalDonation[_senderAddress].add(_senderValue) (REX_Rda.sol#405)

- _reserveRex(_referralAddress,msg.sender,_busd_amount) (REX_Rda.sol#365)

- referralBUSD[_referralAddress] = referralBUSD[_referralAddress].add(_senderValue.mul(mrexRef).div(100)) (REX_Rda.sol#425)

- _reserveRex(_referralAddress,msg.sender,_busd_amount) (REX_Rda.sol#365)

- referrerBalances[_referrer][_donationDay] = referrerBalances[_referrer][_donationDay].add(_referralAmount) (REX_Rda.sol#464)

- _reserveRex(_referralAddress,msg.sender,_busd_amount) (REX_Rda.sol#365)

- referrerTotalBalance[_referralAddress] = referrerTotalBalance[_referralAddress].add(_value) (REX_Rda.sol#491)

- _reserveRex(_referralAddress,msg.sender,_busd_amount) (REX_Rda.sol#365)

- sumOfDonationsOfUnHit = sumOfDonationsOfUnHit.add(_senderValue) (REX_Rda.sol#400)

- _reserveRex(_referralAddress,msg.sender,_busd_amount) (REX_Rda.sol#365)

- toSendToPairBusd = toSendToPairBusd.add(_senderValue.div(10)) (REX_Rda.sol#409)

- _reserveRex(_referralAddress,msg.sender,_busd_amount) (REX_Rda.sol#365)

- uniqueDonatorCount ++ (REX_Rda.sol#476)

- _reserveRex(_referralAddress,msg.sender,_busd_amount) (REX_Rda.sol#365)

- uniqueDonators[uniqueDonatorCount] = _donatorAddress (REX_Rda.sol#475)

- _reserveRex(_referralAddress,msg.sender,_busd_amount) (REX_Rda.sol#365)

- uniqueReferrerCount ++ (REX_Rda.sol#489)

- _reserveRex(_referralAddress,msg.sender,_busd_amount) (REX_Rda.sol#365)

- uniqueReferrers[uniqueReferrerCount] = _referralAddress (REX_Rda.sol#488)

Event emitted after the call(s):

- DonationReceived(_senderAddress,_donationDay,_donationBalance) (REX_Rda.sol#454)

- _reserveRex(_referralAddress,msg.sender,_busd_amount) (REX_Rda.sol#365)

- ReferralAdded(_referralAddress,_senderAddress,amountREX) (REX_Rda.sol#428)

- _reserveRex(_referralAddress,msg.sender,_busd_amount) (REX_Rda.sol#365)

Reentrancy in RexDailyAuction.supplyTrigger() (REX_Rda.sol#298-303):

External calls:

- _dailyDistributionRoutine() (REX_Rda.sol#300)

- msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

External calls sending eth:

- _dailyDistributionRoutine() (REX_Rda.sol#300)

- (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)

- msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

State variables written after the call(s):

- _fillLiquidityPool() (REX_Rda.sol#302)
  - extraLiqBusdSent = extraLiqBusdSent.add(amountBUSD) (REX_Rda.sol#744)
- _fillLiquidityPool() (REX_Rda.sol#302)
  - toSendToPairBusd = toSendToPairBusd.sub(amountBUSD_scope_5) (REX_Rda.sol#796)
- _fillLiquidityPool() (REX_Rda.sol#302)
  - totalLpTokens = LP_TOKEN.balanceOf(address(this)) (REX_Rda.sol#746)

Event emitted after the call(s):

- LiquidityGenerated(_currentRxDay(),amountBUSD,amountREX) (REX_Rda.sol#748)
  - _fillLiquidityPool() (REX_Rda.sol#302)
- LiquidityGenerated(_currentRxDay(),amountBUSD_scope_5,amountREX_scope_4) (REX_Rda.sol#798)
  - _fillLiquidityPool() (REX_Rda.sol#302)

Reentrancy in RexDailyAuction.withdrawLPTokens() (REX_Rda.sol#521-534):

External calls:

- supplyTrigger() (REX_Rda.sol#522)
  - msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

External calls sending eth:

- supplyTrigger() (REX_Rda.sol#522)
  - (success) = recipient.call{value: amount}() (REX_Rda.sol#1312)
  - msg.sender.transfer(refundBNB) (REX_Rda.sol#643)

State variables written after the call(s):

- liquidityBalancesDrawn[msg.sender] = liquidityBalancesDrawn[msg.sender].add(lpTokensPayout) (REX_Rda.sol#530)

Event emitted after the call(s):

- LPtokensWithdrawn(msg.sender,lpTokensPayout) (REX_Rda.sol#533)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4

Variable IUniswapV2Router02.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (REX_Rda.sol#178) is too similar to IUniswapV2Router02.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (REX_Rda.sol#179)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

RexDailyAuction._sendInitialLiquidityToPCS() (REX_Rda.sol#809-853) uses literals with too many digits:

- INITIAL_LIQ_BUSD > 100000E18 (REX_Rda.sol#816)

RexDailyAuction._sendInitialLiquidityToPCS() (REX_Rda.sol#809-853) uses literals with too many digits:

- _startLiquidity = uint256(100000E18) (REX_Rda.sol#818)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

1. Mythril Output

DEX Contract:

Dependence on tx.origin
- SWC ID: 115
- Severity: Low
- Contract: RexDEX
- Function name: `buyStakeFromList(uint256)`
- PC address: 1872
- Estimated Gas Usage: 2882 - 3827

Description

Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
In file: REX_dex.sol:217

Code

```
require(_notContract(msg.sender) && msg.sender == tx.origin, 'DEX: Buyer not an address')
```

Initial State:

Account: [CREATOR], balance: 0x800000000000, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: initRexContract(address), txdata:
0x4527b14d0100010101010101010101affeaffeaffeaffeaffeaffeaffeaffeaffe, value: 0x0
Caller: [CREATOR], function: listStake(address,uint32,uint32,uint256,bytes16), txdata:
0xbf96dd150100010101020101010101010000000000000000000000000000000000000027272727272727272727272727272727272727272727272727272727272727272700000000272727272727272727272727272727272727272727272727272727272727270000

0000272727272727272727270000000000000000000020000000000000000080000000000000000000000000000010
427272727272727272727272727272727, value: 0x0
Caller: [SOMEGUY], function: buyStakeFromList(uint256), txdata:
0x2189e22c0000000000000000000000000000000000000000000000000000000000000000, value: 0x0

Multiple Calls in a Single Transaction
- SWC ID: 113
- Severity: Low
- Contract: RexDEX
- Function name: `buyStakeFromList(uint256)`
- PC address: 2359
- Estimated Gas Usage: 53878 - 347831

Description

Multiple calls are executed in the same transaction.
This call is executed following another call within the same transaction. It is possible that the call never gets executed if a prior call fails permanently. This might be caused intentionally by a malicious callee. If possible, refactor the code such that each transaction only executes one external call or make sure that all callees can be trusted (i.e. they're part of your own codebase).
In file: REX_dex.sol:225

Code

```
MREX_TOKEN.balanceOf(msg.sender)
```

Initial State:

Account: [CREATOR], balance: 0x400000000003, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: initRexContract(address), txdata:
0x4527b14d000000000000000000000000affeaffeaffeaffeaffeaffeaffeaffeaffeaffe, value: 0x0
Caller: [CREATOR], function: listStake(address,uint32,uint32,uint256,bytes16), txdata:
0xbf96dd150101010101010101010101010101000140000101010140800101000000108010140b1b1b1b1b1b1b1b1b1b1b1b
1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1401072beb1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1403b

250400000000000040020170162 1c2d8e8b30408001d18097865a0b80538120008190100000000000000000000000000000b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1, value: 0x0

Caller: [ATTACKER], function: buyStakeFromList(uint256), txdata: 0x2189e22c0000000000000000000000000000000000000000000000000000000000000000, value: 0x0

Dependence on tx.origin

- SWC ID: 115
- Severity: Low
- Contract: RexDEX
- Function name: `buyStakeFromList(uint256)`
- PC address: 2421
- Estimated Gas Usage: 15601 - 107225

Description

Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
In file: REX_dex.sol:225

Code

```
if( MREX_TOKEN.balanceOf(msg.sender) == 0 )
      {
         uint256 _fee = _offer.offerPrice.div(100); // fee = 1% = 1/100 of offer price
         require(BUSD_TOKEN.transferFrom(msg.sender, MARKETING_ADDR, _fee), "DEX: M-Fee transfer failed."); // pay
         require(BUSD_TOKEN.transferFrom(msg.sender, DEVELOPMENT_ADDR, _fee), "DEX: D-Fee transfer failed."); //
pay
      }
```

Initial State:

Account: [CREATOR], balance: 0x8080082000000, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0

Caller: [CREATOR], function: initRexContract(address), txdata:
0x4527b14d01010101010101010101010101affeaffeaffeaffeaffeaffeaffeaffeaffeaffe, value: 0x0
Caller: [CREATOR], function: listStake(address,uint32,uint32,uint256,bytes16), txdata:
0xbf96dd152001010101010101010101010120800801010880290022aaa28800089028880b1b1b1b1b1b1b1b1b1b1b1b
1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1201d6158b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b15105
3e44b1b1b1000020010280010200108000100000000280024002020080000000010810000400200020020800400220800100
4b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1, value: 0x0
Caller: [SOMEGUY], function: buyStakeFromList(uint256), txdata:
0x2189e22c0000000000000000000000000000000000000000000000000000000000000000, value: 0x0

Dependence on tx.origin
- SWC ID: 115
- Severity: Low
- Contract: RexDEX
- Function name: `buyStakeFromList(uint256)`
- PC address: 2709
- Estimated Gas Usage: 19088 - 145794

Description

Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
In file: REX_dex.sol:228

Code

```
require(BUSD_TOKEN.transferFrom(msg.sender, MARKETING_ADDR, _fee), "DEX: M-Fee transfer failed.")
```

Initial State:

Account: [CREATOR], balance: 0x4d006400000000004, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: initRexContract(address), txdata:
0x4527b14d8080808080408040808080deadbeefdeadbeefdeadbeefdeadbeefdeadbeef, value: 0x0

Caller: [ATTACKER], function: listStake(address,uint32,uint32,uint256,bytes16), txdata:
0xbf96dd150101010101000101010101010101010101010101010101010101010101010101b1b1b1b1b1b1b1b1b1b1b1b
1b1b1b1b1b1b1b1b1b1b1b1b1b145400e1fb1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b142c0f
1e1b1b1b10000000000000000000000000000000000000000000000000000000408000000000000000000000000000004
b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1, value: 0x0
Caller: [CREATOR], function: buyStakeFromList(uint256), txdata:
0x2189e22c0000000000000000000000000000000000000000000000000000000000000000, value: 0x0

State access after external call
- SWC ID: 107
- Severity: Low
- Contract: RexDEX
- Function name: `buyStakeFromList(uint256)`
- PC address: 2825
- Estimated Gas Usage: 53878 - 347831

Description

Read of persistent state following external call
The contract account state is accessed after an external call to a fixed address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.
In file: REX_dex.sol:229

Code

```
BUSD_TOKEN
```

Initial State:

Account: [CREATOR], balance: 0x5020000000000003, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x10000400000002, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: initRexContract(address), txdata:
0x4527b14d020410010240008002010808affeaffeaffeaffeaffeaffeaffeaffeaffeaffe, value: 0x0
Caller: [CREATOR], function: listStake(address,uint32,uint32,uint256,bytes16), txdata:
0xbf96dd15000000000000000000000000000000000000000000000000000000000000000000b1b1b1b1b1b1b1b1b1b1b1b

1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b14c9c5768b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b114a9
324800020100b0100200108080200220040040000230a32000000000008040240c164400104022004400008402008022010802
0b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1, value: 0x0
Caller: [SOMEGUY], function: buyStakeFromList(uint256), txdata:
0x2189e22c0000000000000000000000000000000000000000000000000000000000000000, value: 0x0

State access after external call
- SWC ID: 107
- Severity: Low
- Contract: RexDEX
- Function name: `buyStakeFromList(uint256)`
- PC address: 3019
- Estimated Gas Usage: 53878 - 347831

Description

Write to persistent state following external call
The contract account state is accessed after an external call to a fixed address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.
In file: REX_dex.sol:229

Code

```
BUSD_TOKEN.transferFrom(msg.sender, DEVELOPMENT_ADDR, _fee)
```

Initial State:

Account: [CREATOR], balance: 0x5020000000000003, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x10000400000002, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: initRexContract(address), txdata:
0x4527b14d020410010240008002010808affeaffeaffeaffeaffeaffeaffeaffeaffeaffe, value: 0x0
Caller: [CREATOR], function: listStake(address,uint32,uint32,uint256,bytes16), txdata:
0xbf96dd150000000000000000000000000000000000000000000000000000000000000000b1b1b1b1b1b1b1b1b1b1b1b1b
1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b14c9c5768b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b114a9

324800020100b010020010808020022004004000230a32000000008040240c16440010402200440000840200802201080
0b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1, value: 0x0
Caller: [SOMEGUY], function: buyStakeFromList(uint256), txdata:
0x2189e22c0000000000000000000000000000000000000000000000000000000000000000, value: 0x0

Dependence on tx.origin
- SWC ID: 115
- Severity: Low
- Contract: RexDEX
- Function name: `buyStakeFromList(uint256)`
- PC address: 3079
- Estimated Gas Usage: 21629 - 183417

Description

Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
In file: REX_dex.sol:229

Code

```
require(BUSD_TOKEN.transferFrom(msg.sender, DEVELOPMENT_ADDR, _fee), "DEX: D-Fee transfer failed.")
```

Initial State:

Account: [CREATOR], balance: 0x32081098583183483, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x1, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: initRexContract(address), txdata:
0x4527b14d0000010101010101010101aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa, value: 0x0
Caller: [SOMEGUY], function: listStake(address,uint32,uint32,uint256,bytes16), txdata:
0xbf96dd150000000000000000000000000000000000000000000000000000000000000001b1b1b1b1b1b1b1b1b1b1b1b
1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1f7299e14b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b10000
0008b1b1b1000000000000000000000000000000000000000000000000000000000000800000000000000000000000000000
4b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1, value: 0x0

Caller: [ATTACKER], function: buyStakeFromList(uint256), txdata:
0x2189e22c0000000000000000000000000000000000000000000000000000000000000000, value: 0x0

State access after external call
- SWC ID: 107
- Severity: Low
- Contract: RexDEX
- Function name: `buyStakeFromList(uint256)`
- PC address: 3197
- Estimated Gas Usage: 53878 - 347831

Description

Read of persistent state following external call
The contract account state is accessed after an external call to a fixed address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.
In file: REX_dex.sol:233

Code

```
BUSD_TOKEN
```

Initial State:

Account: [CREATOR], balance: 0x5020000000000003, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x10000400000002, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: initRexContract(address), txdata:
0x4527b14d02041001024000800201080 8affeaffeaffeaffeaffeaffeaffeaffeaffeaffe, value: 0x0
Caller: [CREATOR], function: listStake(address,uint32,uint32,uint256,bytes16), txdata:
0xbf96dd150000000000000000000000000000000000000000000000000000000000000000b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b14c9c5768b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b114a9324800020100b0100200108080200220040040002 30a320000000008040240c16440010402200440000840200802201080 20b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1, value: 0x0
Caller: [SOMEGUY], function: buyStakeFromList(uint256), txdata:
0x2189e22c0000000000000000000000000000000000000000000000000000000000000000, value: 0x0

State access after external call

- SWC ID: 107
- Severity: Low
- Contract: RexDEX
- Function name: `buyStakeFromList(uint256)`
- PC address: 3262
- Estimated Gas Usage: 53878 - 347831

Description

Read of persistent state following external call
The contract account state is accessed after an external call to a fixed address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.
In file: REX_dex.sol:233

Code

```
_offer.staker
```

Initial State:

Account: [CREATOR], balance: 0x5020000000000003, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x10000400000002, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: initRexContract(address), txdata: 0x4527b14d0204100102400080020108 08affeaffeaffeaffeaffeaffeaffeaffeaffeaffe, value: 0x0
Caller: [CREATOR], function: listStake(address,uint32,uint32,uint256,bytes16), txdata: 0xbf96dd150000000000000000000000000000000000000000000000000000000000000000b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b14c9c5768b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b114a9324800020100b010020010808020022004004000230a320000000008040240c164400104022004400008402008022010802 0b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1, value: 0x0
Caller: [SOMEGUY], function: buyStakeFromList(uint256), txdata: 0x2189e22c0000000000000000000000000000000000000000000000000000000000000000, value: 0x0

State access after external call

- SWC ID: 107
- Severity: Low
- Contract: RexDEX
- Function name: `buyStakeFromList(uint256)`
- PC address: 3296
- Estimated Gas Usage: 53878 - 347831

Description

Read of persistent state following external call
The contract account state is accessed after an external call to a fixed address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.
In file: REX_dex.sol:233

Code

```
_offer.offerPrice
```

Initial State:

Account: [CREATOR], balance: 0x5020000000000003, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x10000400000002, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: initRexContract(address), txdata: 0x4527b14d020410010240008002010808affeaffeaffeaffeaffeaffeaffeaffeaffeaffe, value: 0x0
Caller: [CREATOR], function: listStake(address,uint32,uint32,uint256,bytes16), txdata: 0xbf96dd150000000000000000000000000000000000000000000000000000000000000000b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b14c9c5768b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b114a9324800020100b010020010808020022004004000230a320000000008040240c1644001040220044000084020080220108020b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1, value: 0x0
Caller: [SOMEGUY], function: buyStakeFromList(uint256), txdata: 0x2189e22c0000000000000000000000000000000000000000000000000000000000000000, value: 0x0

State access after external call
- SWC ID: 107
- Severity: Low

- Contract: RexDEX
- Function name: `buyStakeFromList(uint256)`
- PC address: 3411
- Estimated Gas Usage: 53878 - 347831

Description

Write to persistent state following external call
The contract account state is accessed after an external call to a fixed address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.
In file: REX_dex.sol:233

Code

```
BUSD_TOKEN.transferFrom(msg.sender, _offer.staker, _offer.offerPrice)
```

Initial State:

Account: [CREATOR], balance: 0x5020000000000003, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x10000400000002, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: initRexContract(address), txdata:
0x4527b14d02041001024000800201080 8affeaffeaffeaffeaffeaffeaffeaffeaffeaffe, value: 0x0
Caller: [CREATOR], function: listStake(address,uint32,uint32,uint256,bytes16), txdata:
0xbf96dd150000000000000000000000000000000000000000000000000000000000000000b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b14c9c5768b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b114a93248000201 00b0100200108080200220040040002 30a320000000008040240c16440010402200440000840200802201080 20b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1, value: 0x0
Caller: [SOMEGUY], function: buyStakeFromList(uint256), txdata:
0x2189e22c0000000000000000000000000000000000000000000000000000000000000000, value: 0x0

Dependence on tx.origin
- SWC ID: 115
- Severity: Low
- Contract: RexDEX
- Function name: `buyStakeFromList(uint256)`

- PC address: 3471
- Estimated Gas Usage: 19796 - 146832

Description

Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
In file: REX_dex.sol:233

Code

```
require(BUSD_TOKEN.transferFrom(msg.sender, _offer.staker, _offer.offerPrice), "DEX: Transfer of BUSD failed.")
```

Initial State:

Account: [CREATOR], balance: 0x2, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: initRexContract(address), txdata: 0x4527b14d000000000000000000000000affeaffeaffeaffeaffeaffeaffeaffeaffeaffe, value: 0x0
Caller: [CREATOR], function: listStake(address,uint32,uint32,uint256,bytes16), txdata: 0xbf96dd1500000000000000000000000000000000000000000000000000000000000000000001b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1dffff250b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1000010 00b1b1b100004000000000000000000000000000000000000000000000000000000000004080000000000000000000104b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1, value: 0x0
Caller: [CREATOR], function: buyStakeFromList(uint256), txdata: 0x2189e22c0000000000000000000000000000000000000000000000000000000000000000, value: 0x0

State access after external call
- SWC ID: 107
- Severity: Low
- Contract: RexDEX
- Function name: `buyStakeFromList(uint256)`
- PC address: 3589
- Estimated Gas Usage: 53878 - 347831

Description

Read of persistent state following external call

The contract account state is accessed after an external call to a fixed address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.

In file: REX_dex.sol:235

Code

```
gl.totalActiveOffers
```

Initial State:

Account: [CREATOR], balance: 0x5020000000000003, nonce:0, storage:{}

Account: [ATTACKER], balance: 0x10000400000002, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0

Caller: [CREATOR], function: initRexContract(address), txdata: 0x4527b14d020410010240008002010808affeaffeaffeaffeaffeaffeaffeaffeaffeaffe, value: 0x0

Caller: [CREATOR], function: listStake(address,uint32,uint32,uint256,bytes16), txdata: 0xbf96dd150000000000000000000000000000000000000000000000000000000000000000b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b14c9c5768b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b114a9324800020100b0100200108080200220040040002 30a32000000000804 0240c1644001040220044000084020080220108020b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1, value: 0x0

Caller: [SOMEGUY], function: buyStakeFromList(uint256), txdata: 0x2189e22c0000000000000000000000000000000000000000000000000000000000000000, value: 0x0

State access after external call

- SWC ID: 107
- Severity: Low
- Contract: RexDEX
- Function name: `buyStakeFromList(uint256)`
- PC address: 3613
- Estimated Gas Usage: 53878 - 347831

Description

Read of persistent state following external call

The contract account state is accessed after an external call to a fixed address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.

In file: REX_dex.sol:235

Code

```
gl.totalActiveOffers
```

Initial State:

Account: [CREATOR], balance: 0x5020000000000003, nonce:0, storage:{}

Account: [ATTACKER], balance: 0x10000400000002, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0

Caller: [CREATOR], function: initRexContract(address), txdata:
0x4527b14d020410010240008002010808affeaffeaffeaffeaffeaffeaffeaffeaffe, value: 0x0

Caller: [CREATOR], function: listStake(address,uint32,uint32,uint256,bytes16), txdata:
0xbf96dd150000000000000000000000000000000000000000000000000000000000000b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b14c9c5768b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b114a93248000201000b0100200108080200220040040002300a320000000008040240c16440010402200440000840200802201080200b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1, value: 0x0

Caller: [SOMEGUY], function: buyStakeFromList(uint256), txdata:
0x2189e22c0000000000000000000000000000000000000000000000000000000000000000, value: 0x0

State access after external call

- SWC ID: 107
- Severity: Low
- Contract: RexDEX
- Function name: `buyStakeFromList(uint256)`
- PC address: 3636
- Estimated Gas Usage: 53878 - 347831

Description

Write to persistent state following external call

The contract account state is accessed after an external call to a fixed address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.
In file: REX_dex.sol:235

Code

```
gl.totalActiveOffers = gl.totalActiveOffers >= 1 ? gl.totalActiveOffers.sub(1) : 0
```

Initial State:

Account: [CREATOR], balance: 0x5020000000000003, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x10000400000002, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: initRexContract(address), txdata:
0x4527b14d020410010240008002010808affeaffeaffeaffeaffeaffeaffeaffeaffe, value: 0x0
Caller: [CREATOR], function: listStake(address,uint32,uint32,uint256,bytes16), txdata:
0xbf96dd150000000000000000000000000000000000000000000000000000000000000000b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b14c9c5768b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b114a9324800020100b010020010808020022004004000230a320000000008040240c16440010402200440000840200802201080201b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1, value: 0x0
Caller: [SOMEGUY], function: buyStakeFromList(uint256), txdata:
0x2189e22c0000000000000000000000000000000000000000000000000000000000000000, value: 0x0

State access after external call
- SWC ID: 107
- Severity: Low
- Contract: RexDEX
- Function name: `buyStakeFromList(uint256)`
- PC address: 3648
- Estimated Gas Usage: 53878 - 347831

Description

Read of persistent state following external call

The contract account state is accessed after an external call to a fixed address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.
In file: REX_dex.sol:236

Code

```
gl.totalFulfilledOffers
```

Initial State:

Account: [CREATOR], balance: 0x5020000000000003, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x10000400000002, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: initRexContract(address), txdata:
0x4527b14d020410010240008002010808affeaffeaffeaffeaffeaffeaffeaffeaffe, value: 0x0
Caller: [CREATOR], function: listStake(address,uint32,uint32,uint256,bytes16), txdata:
0xbf96dd150000000000000000000000000000000000000000000000000000000000000000b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b14c9c5768b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b114a9324800020100b010020010808020022004004000230a320000000008040240c16440010402200440000840200802201080201b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1, value: 0x0
Caller: [SOMEGUY], function: buyStakeFromList(uint256), txdata:
0x2189e22c0000000000000000000000000000000000000000000000000000000000000000, value: 0x0

State access after external call
- SWC ID: 107
- Severity: Low
- Contract: RexDEX
- Function name: `buyStakeFromList(uint256)`
- PC address: 3670
- Estimated Gas Usage: 53878 - 347831

Description

Write to persistent state following external call

The contract account state is accessed after an external call to a fixed address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.
In file: REX_dex.sol:236

Code

```
gl.totalFulfilledOffers = gl.totalFulfilledOffers.add(1)
```

Initial State:

Account: [CREATOR], balance: 0x5020000000000003, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x10000400000002, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: initRexContract(address), txdata:
0x4527b14d020410010240008002010808affeaffeaffeaffeaffeaffeaffeaffeaffe, value: 0x0
Caller: [CREATOR], function: listStake(address,uint32,uint32,uint256,bytes16), txdata:
0xbf96dd150000000000000000000000000000000000000000000000000000000000000000b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b14c9c5768b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b114a9324800020100b010020010808020022004004000230a32000000000804024 0c164400104022004400008402008022010802 0b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1, value: 0x0
Caller: [SOMEGUY], function: buyStakeFromList(uint256), txdata:
0x2189e22c0000000000000000000000000000000000000000000000000000000000000000, value: 0x0

State access after external call
- SWC ID: 107
- Severity: Low
- Contract: RexDEX
- Function name: `buyStakeFromList(uint256)`
- PC address: 3679
- Estimated Gas Usage: 53878 - 347831

Description

Read of persistent state following external call

The contract account state is accessed after an external call to a fixed address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.
In file: REX_dex.sol:237

Code

```
_offer.offerPrice
```

Initial State:

Account: [CREATOR], balance: 0x5020000000000003, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x10000400000002, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: initRexContract(address), txdata:
0x4527b14d020410010240008002010808affeaffeaffeaffeaffeaffeaffeaffeaffe, value: 0x0
Caller: [CREATOR], function: listStake(address,uint32,uint32,uint256,bytes16), txdata:
0xbf96dd150000000000000000000000000000000000000000000000000000000000000000b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b14c9c5768b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b114a9324800020100b010020010808020022004004000230a32000000008040240c16440010402200440000840200802201080209b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1, value: 0x0
Caller: [SOMEGUY], function: buyStakeFromList(uint256), txdata:
0x2189e22c0000000000000000000000000000000000000000000000000000000000000000, value: 0x0

State access after external call
- SWC ID: 107
- Severity: Low
- Contract: RexDEX
- Function name: `buyStakeFromList(uint256)`
- PC address: 3685
- Estimated Gas Usage: 53878 - 347831

Description

Read of persistent state following external call

The contract account state is accessed after an external call to a fixed address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.
In file: REX_dex.sol:237

Code

```
gl.totalTradingVolume
```

Initial State:

Account: [CREATOR], balance: 0x5020000000000003, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x10000400000002, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: initRexContract(address), txdata:
0x4527b14d020410010240008002010808affeaffeaffeaffeaffeaffeaffeaffeaffe, value: 0x0
Caller: [CREATOR], function: listStake(address,uint32,uint32,uint256,bytes16), txdata:
0xbf96dd150000000000000000000000000000000000000000000000000000000000000000b1b1b1b1b1b1b1b1b1b1b1b
1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b14c9c5768b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b114a9
324800020100b010020010808020022004004000230a320000000008040240c16440010402200440000840200802201080 2
0b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1, value: 0x0
Caller: [SOMEGUY], function: buyStakeFromList(uint256), txdata:
0x2189e22c0000000000000000000000000000000000000000000000000000000000000000, value: 0x0

State access after external call
- SWC ID: 107
- Severity: Low
- Contract: RexDEX
- Function name: `buyStakeFromList(uint256)`
- PC address: 3707
- Estimated Gas Usage: 53878 - 347831

Description

Write to persistent state following external call

The contract account state is accessed after an external call to a fixed address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.

In file: REX_dex.sol:237

Code

```
gl.totalTradingVolume = gl.totalTradingVolume.add(_offer.offerPrice)
```

Initial State:

Account: [CREATOR], balance: 0x5020000000000003, nonce:0, storage:{}

Account: [ATTACKER], balance: 0x10000400000002, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0

Caller: [CREATOR], function: initRexContract(address), txdata: 0x4527b14d0204100102400080020108 08affeaffeaffeaffeaffeaffeaffeaffeaffeaffe, value: 0x0

Caller: [CREATOR], function: listStake(address,uint32,uint32,uint256,bytes16), txdata: 0xbf96dd1500000000000000000000000000000000000000000000000000000000000000000000b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b14c9c5768b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b114a9324800020100b0100200108080200220040040000230a3200000000080402 40c16440010402200440000840200802201 08020b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1, value: 0x0

Caller: [SOMEGUY], function: buyStakeFromList(uint256), txdata: 0x2189e22c0000000000000000000000000000000000000000000000000000000000000000, value: 0x0

State access after external call

- SWC ID: 107
- Severity: Low
- Contract: RexDEX
- Function name: `buyStakeFromList(uint256)`
- PC address: 3714
- Estimated Gas Usage: 53878 - 347831

Description

Read of persistent state following external call

The contract account state is accessed after an external call to a fixed address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.
In file: REX_dex.sol:239

Code

```
REX_CONTRACT
```

Initial State:

Account: [CREATOR], balance: 0x5020000000000003, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x10000400000002, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: initRexContract(address), txdata:
0x4527b14d0204100102400080020108 08affeaffeaffeaffeaffeaffeaffeaffeaffeaffe, value: 0x0
Caller: [CREATOR], function: listStake(address,uint32,uint32,uint256,bytes16), txdata:
0xbf96dd150000000000000000000000000000000000000000000000000000000000000000b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b14c9c5768b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b114a9324800020100b0100200108080200220040040002 30a320000000008040240c16440010402200440000840200802201080 20b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1, value: 0x0
Caller: [SOMEGUY], function: buyStakeFromList(uint256), txdata:
0x2189e22c0000000000000000000000000000000000000000000000000000000000000000, value: 0x0

State access after external call
- SWC ID: 107
- Severity: Low
- Contract: RexDEX
- Function name: `buyStakeFromList(uint256)`
- PC address: 3778
- Estimated Gas Usage: 53878 - 347831

Description

Read of persistent state following external call

The contract account state is accessed after an external call to a fixed address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.
In file: REX_dex.sol:239

Code

```
_offer.stakeID
```

Initial State:

Account: [CREATOR], balance: 0x5020000000000003, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x10000400000002, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: initRexContract(address), txdata:
0x4527b14d020410010240008002010808affeaffeaffeaffeaffeaffeaffeaffeaffe, value: 0x0
Caller: [CREATOR], function: listStake(address,uint32,uint32,uint256,bytes16), txdata:
0xbf96dd150000000000000000000000000000000000000000000000000000000000000000b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b14c9c5768b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b114a9324800020100b010020010808020022004004000230a320000000008040240c16440010402200440000840200802201080209b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1, value: 0x0
Caller: [SOMEGUY], function: buyStakeFromList(uint256), txdata:
0x2189e22c0000000000000000000000000000000000000000000000000000000000000000, value: 0x0

State access after external call
- SWC ID: 107
- Severity: Low
- Contract: RexDEX
- Function name: `buyStakeFromList(uint256)`
- PC address: 3796
- Estimated Gas Usage: 53878 - 347831

Description

Read of persistent state following external call

The contract account state is accessed after an external call to a fixed address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.
In file: REX_dex.sol:239

Code

```
_offer.staker
```

Initial State:

Account: [CREATOR], balance: 0x5020000000000003, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x10000400000002, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: initRexContract(address), txdata:
0x4527b14d020410010240008002010808affeaffeaffeaffeaffeaffeaffeaffeaffe, value: 0x0
Caller: [CREATOR], function: listStake(address,uint32,uint32,uint256,bytes16), txdata:
0xbf96dd150000000000000000000000000000000000000000000000000000000000000000b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b14c9c5768b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b114a9324800020100b010020010808020022004004000230a320000000008040240c16440010402200440000840200802201080209b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1, value: 0x0
Caller: [SOMEGUY], function: buyStakeFromList(uint256), txdata:
0x2189e22c0000000000000000000000000000000000000000000000000000000000000000, value: 0x0

External Call To User-Supplied Address
- SWC ID: 107
- Severity: Low
- Contract: RexDEX
- Function name: `buyStakeFromList(uint256)`
- PC address: 3960
- Estimated Gas Usage: 53878 - 347831

Description

A call to a user-supplied address is executed.
An external message call to an address specified by the caller is executed. Note that the callee account might contain arbitrary code and could re-enter any function within this contract. Reentering the contract in an intermediate state may lead

to unexpected behaviour. Make sure that no state modifications are executed after this call and/or reentrancy guards are in place.
In file: REX_dex.sol:239

Code

```
REX_CONTRACT.createBoughtStake(_offer.stakeID, _offer.staker, msg.sender)
```

Initial State:

Account: [CREATOR], balance: 0x1, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x800000001000, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: initRexContract(address), txdata: 0x4527b14d200180080808010820100101deadbeefdeadbeefdeadbeefdeadbeefdeadbeef, value: 0x0
Caller: [ATTACKER], function: listStake(address,uint32,uint32,uint256,bytes16), txdata: 0xbf96dd150120102001018001000180800102002004040000001000008002000000400020b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b124d1b230b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1840302e0060001129590801000000640400000000000000000000000000000000000000014020020101100280801040000440800b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1, value: 0x0
Caller: [ATTACKER], function: buyStakeFromList(uint256), txdata: 0x2189e22c0000000000000000000000000000000000000000000000000000000000000000, value: 0x0

State access after external call
- SWC ID: 107
- Severity: Low
- Contract: RexDEX
- Function name: `buyStakeFromList(uint256)`
- PC address: 3960
- Estimated Gas Usage: 53878 - 347831

Description

Write to persistent state following external call
The contract account state is accessed after an external call to a fixed address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.

In file: REX_dex.sol:239

Code

```
REX_CONTRACT.createBoughtStake(_offer.stakeID, _offer.staker, msg.sender)
```

Initial State:

Account: [CREATOR], balance: 0x5020000000000003, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x10000400000002, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: initRexContract(address), txdata:
0x4527b14d020410010240008002010808affeaffeaffeaffeaffeaffeaffeaffeaffeaffe, value: 0x0
Caller: [CREATOR], function: listStake(address,uint32,uint32,uint256,bytes16), txdata:
0xbf96dd150000000000000000000000000000000000000000000000000000000000000000b1b1b1b1b1b1b1b1b1b1b1b
1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b14c9c5768b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b114a9
324800020100b010020010808020022004004000230a320000000008040240c164400104022004400008402008022010802
0b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1, value: 0x0
Caller: [SOMEGUY], function: buyStakeFromList(uint256), txdata:
0x2189e22c0000000000000000000000000000000000000000000000000000000000000000, value: 0x0

State access after external call
- SWC ID: 107
- Severity: Medium
- Contract: RexDEX
- Function name: `buyStakeFromList(uint256)`
- PC address: 3986
- Estimated Gas Usage: 53878 - 347831

Description

Read of persistent state following external call
The contract account state is accessed after an external call to a user defined address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.
In file: REX_dex.sol:241

Code

```
_offer.offerPrice
```

Initial State:

Account: [CREATOR], balance: 0x3000201000400062, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x20001c600420102, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: initRexContract(address), txdata:
0x4527b14d40000200400480000100080deadbeefdeadbeefdeadbeefdeadbeefdeadbeef, value: 0x0
Caller: [ATTACKER], function: listStake(address,uint32,uint32,uint256,bytes16), txdata:
0xbf96dd150000000000000000000000000000000000000000000000000800000000000000000b1b1b1b1b1b1b1b1b1b1b1b
1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b188d424d8b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b11038
6a8a1040400e0502080180400008002800000000000000001000000000000000000020220100402104001000102010001 4
0b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1, value: 0x0
Caller: [ATTACKER], function: buyStakeFromList(uint256), txdata:
0x2189e22c0000000000000000000000000000000000000000000000000000000000000000, value: 0x0

State access after external call
- SWC ID: 107
- Severity: Medium
- Contract: RexDEX
- Function name: `buyStakeFromList(uint256)`
- PC address: 4017
- Estimated Gas Usage: 53878 - 347831

Description

Read of persistent state following external call
The contract account state is accessed after an external call to a user defined address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.
In file: REX_dex.sol:241

Code

```
_offer.staker
```

Initial State:

Account: [CREATOR], balance: 0x3000201000400062, nonce:0, storage:{}

Account: [ATTACKER], balance: 0x20001c600420102, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0

Caller: [CREATOR], function: initRexContract(address), txdata:
0x4527b14d40000200400480000 1000080deadbeefdeadbeefdeadbeefdeadbeefdeadbeef, value: 0x0

Caller: [ATTACKER], function: listStake(address,uint32,uint32,uint256,bytes16), txdata:
0xbf96dd150000000000000000000000000000000000000000000000000800000000000000000b1b1b1b1b1b1b1b1b1b1b1b
1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b188d424d8b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b11038
6a8a1040400e0502080180400008002800000000000000001000000000000000000020220100402104001000102010001 4
0b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1, value: 0x0

Caller: [ATTACKER], function: buyStakeFromList(uint256), txdata:
0x2189e22c0000000000000000000000000000000000000000000000000000000000000000, value: 0x0

State access after external call

- SWC ID: 107
- Severity: Medium
- Contract: RexDEX
- Function name: `buyStakeFromList(uint256)`
- PC address: 4109
- Estimated Gas Usage: 53878 - 347831

Description

Read of persistent state following external call
The contract account state is accessed after an external call to a user defined address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.
In file: REX_dex.sol:241

Code

```
_offer.stakeID
```

```

### Initial State:

Account: [CREATOR], balance: 0x3000201000400062, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x20001c600420102, nonce:0, storage:{}

### Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: initRexContract(address), txdata:
0x4527b14d40000200400480000100080deadbeefdeadbeefdeadbeefdeadbeefdeadbeef, value: 0x0
Caller: [ATTACKER], function: listStake(address,uint32,uint32,uint256,bytes16), txdata:
0xbf96dd150000000000000000000000000000000000000000000000000800000000000000000b1b1b1b1b1b1b1b1b1b1b1b
1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b188d424d8b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b11038
6a8a1040400e0502080180400008002800000000000000001000000000000000000020220100402104001000102010001 4
0b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1, value: 0x0
Caller: [ATTACKER], function: buyStakeFromList(uint256), txdata:
0x2189e22c0000000000000000000000000000000000000000000000000000000000000000, value: 0x0

## External Call To User-Supplied Address
- SWC ID: 107
- Severity: Low
- Contract: RexDEX
- Function name: `revokeOffer(uint256)`
- PC address: 4799
- Estimated Gas Usage: 25165 - 122438

### Description

A call to a user-supplied address is executed.
An external message call to an address specified by the caller is executed. Note that the callee account might contain arbitrary code and could re-enter any function within this contract. Reentering the contract in an intermediate state may lead to unexpected behaviour. Make sure that no state modifications are executed after this call and/or reentrancy guards are in place.
In file: REX_dex.sol:259

### Code

```
REX_CONTRACT.restoreStake(msg.sender, _offer.stakeID)
```
```

Initial State:

Account: [CREATOR], balance: 0x82000000001, nonce:0, storage:{}

Account: [ATTACKER], balance: 0x400003, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0

Caller: [CREATOR], function: initRexContract(address), txdata:
0x4527b14d010001010101010101010101deadbeefdeadbeefdeadbeefdeadbeefdeadbeef, value: 0x0

Caller: [ATTACKER], function: listStake(address,uint32,uint32,uint256,bytes16), txdata:
0xbf96dd150101010101010101010101affeaffeaffeaffeaffeaffeaffeaffeaffeb1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b
1b1b1b1b1b1b1b1b1b1b100000000b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1b1004d0000000000
0000000000000000020000000000000000000020000000000000000040000400000004010000020000000000220b1b1b1b1b
1b1b1b1b1b1b1b1b1b1b1b1b1, value: 0x0

Caller: [CREATOR], function: revokeOffer(uint256), txdata:
0x3a6a65ce0000000000000000000000000000000000000000000000000000000000000000, value: 0x0

State access after external call

- SWC ID: 107
- Severity: Medium
- Contract: RexDEX
- Function name: `revokeOffer(uint256)`
- PC address: 4884
- Estimated Gas Usage: 25165 - 122438

Description

Read of persistent state following external call
The contract account state is accessed after an external call to a user defined address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.
In file: REX_dex.sol:261

Code

```
_offer.stakeID
```

Initial State:

Account: [CREATOR], balance: 0x20000000000018, nonce:0, storage:{}

Account: [ATTACKER], balance: 0x20000000000800, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0

Caller: [CREATOR], function: initRexContract(address), txdata:
0x4527b14d8080808080808080808080deadbeefdeadbeefdeadbeefdeadbeefdeadbeef, value: 0x0

Caller: [ATTACKER], function: listStake(address,uint32,uint32,uint256,bytes16), txdata:
0xbf96dd150080000000000000000000000affeaffeaffeaffeaffeaffeaffeaffeaffe4545454545454545454545454545454545
5454545454545454545450000400045454545454545454545454545454545454545454545454545454500000000000000000
0000000000000000000000000000000000020120020102010010800040000480010802041000040000401020204545454545
5454545454545454545454545, value: 0x0

Caller: [CREATOR], function: revokeOffer(uint256), txdata:
0x3a6a65ce0000000000000000000000000000000000000000000000000000000000000000, value: 0x0

Airdrop Contract:

Exception State

- SWC ID: 110
- Severity: Medium
- Contract: RexAirdrop
- Function name: `initClaimables(address[],uint256[])`
- PC address: 2015
- Estimated Gas Usage: 9472 - 36009

Description

An assertion violation was triggered.
It is possible to trigger an assertion violation. Note that Solidity assert() statements should only be used to check invariants. Review the transaction trace generated for this issue and either make sure your program logic is correct, or use require() instead of assert() if your goal is to constrain user inputs or enforce preconditions. Remember to validate inputs from both callers (for instance, via passed arguments) and callees (for instance, via return values).
In file: REX_AIRDROP.sol:242

Code

```
function _notContract(address _addr) private view returns (bool) {
      uint32 size;
      assembly { size := extcodesize(_addr) }
      return (size == 0);
   }
```

Initial State:

Account: [CREATOR], balance: 0x4280000000020095, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: initClaimables(address[],uint256[]), txdata:
0x091c00a8000000000000000000000000000000000000000000000000000000000000004000000000000000000000000000000000000000000000000000000000000000600000000000000000000000000000000000000000000000000000000000000010000000000000000000000000000000000000000000000000000000000000001, value: 0x0

TREX Contract:

Dependence on tx.origin
- SWC ID: 115
- Severity: Low

- Contract: TREX
- Function name: `buyOneTREX()`
- PC address: 6582
- Estimated Gas Usage: 1033 - 1128

Description

Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
In file: REX_TREX.sol:802

Code

```
require(! isContract(msg.sender) && msg.sender == tx.origin, "TREX: Contracts cannot buy!")
```

Initial State:

Account: [CREATOR], balance: 0x4000101020401, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [SOMEGUY], function: buyOneTREX(), txdata: 0x90eb1210, value: 0x0

Dependence on tx.origin
- SWC ID: 115
- Severity: Low
- Contract: TREX
- Function name: `mint(uint256)`
- PC address: 8607
- Estimated Gas Usage: 1244 - 1669

Description

Use of tx.origin as a part of authorization control.

The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
In file: REX_TREX.sol:431

Code

```
require(_owner == _msgSender(), "Ownable: caller is not the owner")
```

Initial State:

Account: [CREATOR], balance: 0x10000c3213, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x1, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [ATTACKER], function: mint(uint256), txdata: 0xa0712d6800000000000000000000000000000000000000000000000000000000000000000, value: 0x0

Dependence on tx.origin
- SWC ID: 115
- Severity: Low
- Contract: TREX
- Function name: `initClaimables(address[],uint32[])`
- PC address: 9080
- Estimated Gas Usage: 1894 - 7683

Description

Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
In file: REX_TREX.sol:431

Code

require(_owner == _msgSender(), "Ownable: caller is not the owner")

```

### Initial State:

Account: [CREATOR], balance: 0x1000000801005, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}

### Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [ATTACKER], function: initClaimables(address[],uint32[]), txdata: 0xae18c09d00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000, value: 0x0

## Exception State
- SWC ID: 110
- Severity: Medium
- Contract: TREX
- Function name: `initClaimables(address[],uint32[])`
- PC address: 9394
- Estimated Gas Usage: 9780 - 36647

### Description

An assertion violation was triggered.
It is possible to trigger an assertion violation. Note that Solidity assert() statements should only be used to check invariants. Review the transaction trace generated for this issue and either make sure your program logic is correct, or use require() instead of assert() if your goal is to constrain user inputs or enforce preconditions. Remember to validate inputs from both callers (for instance, via passed arguments) and callees (for instance, via return values).
In file: REX_TREX.sol:46

### Code

```
function add(uint256 a, uint256 b) internal pure returns (uint256) {
        uint256 c = a + b;
        require(c >= a, "SafeMath: addition overflow");

        return c;
    }
```
```

Initial State:

Account: [CREATOR], balance: 0x410000000000, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: initClaimables(address[],uint32[]), txdata:
0xae18c09d00000000000000000000000000000000000000000000000000000000000000400000000000000000000000000000000000000000000000000000000000000600000000000000000000000000000000000000000000000000000000000000000001000000000000000000000000000000000000000000000000000000000000000000, value: 0x0

Dependence on tx.origin
- SWC ID: 115
- Severity: Low
- Contract: TREX
- Function name: `transferOwnership(address)`
- PC address: 11266
- Estimated Gas Usage: 1224 - 1649

Description

Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
In file: REX_TREX.sol:431

Code

```
require(_owner == _msgSender(), "Ownable: caller is not the owner")
```

Initial State:

Account: [CREATOR], balance: 0x1003, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x1, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0

Caller: [ATTACKER], function: transferOwnership(address), txdata:
0xf2fde38b0000000000000000000000000000000000000000000000000000000000000000, value: 0x0

Dependence on tx.origin

- SWC ID: 115
- Severity: Low
- Contract: TREX
- Function name: `approve(address,uint256)`
- PC address: 11455
- Estimated Gas Usage: 451 - 546

Description

Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
In file: REX_TREX.sol:698

Code

```
require(owner != address(0), "BEP20: approve from the zero address")
```

Initial State:

Account: [CREATOR], balance: 0x1112002a02202, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x1, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [ATTACKER], function: approve(address,uint256), txdata:
0x095ea7b3000000000000000000000000deadbeefdeadbeefdeadbeefdeadbeefdeadbeef0000000000000000000000000000000000000000000000000000000000000000, value: 0x0

Dependence on tx.origin

- SWC ID: 115
- Severity: Low
- Contract: TREX

- Function name: `transferFrom(address,address,uint256)`
- PC address: 11589
- Estimated Gas Usage: 15660 - 62606

Description

Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
In file: REX_TREX.sol:699

Code

```
require(spender != address(0), "BEP20: approve to the zero address")
```

Initial State:

Account: [CREATOR], balance: 0x100c0822020, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x80020010, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [ATTACKER], function: transferFrom(address,address,uint256), txdata: 0x23b872dd800000000000000000000001000000000100000000000000000000000000000000000000000000000000000000000000000000000000000000000000010000000000000000000001000000000000000000000000000000000000000000000000000000000000000000000000, value: 0x0

Dependence on tx.origin
- SWC ID: 115
- Severity: Low
- Contract: TREX
- Function name: `transfer(address,uint256)`
- PC address: 11958
- Estimated Gas Usage: 427 - 522

Description

Use of tx.origin as a part of authorization control.

The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
In file: REX_TREX.sol:640

Code

```
require(sender != address(0), "BEP20: transfer from the zero address")
```

Initial State:

Account: [CREATOR], balance: 0x200c77b285, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [SOMEGUY], function: transfer(address,uint256), txdata: 0xa9059cbb000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000, value: 0x0

Dependence on predictable environment variable
- SWC ID: 116
- Severity: Low
- Contract: TREX
- Function name: `noOfTREXSellable(address)`
- PC address: 13083
- Estimated Gas Usage: 4317 - 5972

Description

A control flow decision is made based on The block.timestamp environment variable.
The block.timestamp environment variable is used to determine a control flow decision. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.
In file: REX_TREX.sol:182

Code

```
require(c >= a)
```

Initial State:

Account: [CREATOR], balance: 0x200000262c01025, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: noOfTREXSellable(address), txdata:
0x4d037de8000000000000000000000000000000000000000000000000000000000010400100, value: 0x0

Dependence on tx.origin
- SWC ID: 115
- Severity: Low
- Contract: TREX
- Function name: `mint(uint256)`
- PC address: 13239
- Estimated Gas Usage: 1337 - 1762

Description

Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
In file: REX_TREX.sol:658

Code

```
require(account != address(0), "BEP20: mint to the zero address")
```

Initial State:

Account: [CREATOR], balance: 0x3, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: mint(uint256), txdata:
0xa0712d6800000000000000000000000000000000000000000000000000000000000000000, value: 0x0

REX Contract:

Dependence on tx.origin
- SWC ID: 115
- Severity: Low
- Contract: RexToken
- Function name: `manualSnapshotOneDay()`
- PC address: 9894
- Estimated Gas Usage: 1084 - 1179

Description

Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
In file: REX_REX.sol:618

Code

```
require(_notContract(msg.sender) && msg.sender == tx.origin, 'REX: Not an address')
```

Initial State:

Account: [CREATOR], balance: 0x5280000041004, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: manualSnapshotOneDay(), txdata: 0x14229943, value: 0x0

Dependence on predictable environment variable

- SWC ID: 116
- Severity: Low
- Contract: RexToken
- Function name: `manualSnapshotOneDay()`
- PC address: 10025
- Estimated Gas Usage: 2839 - 2934

Description

A control flow decision is made based on The block.timestamp environment variable.
The block.timestamp environment variable is used to determine a control flow decision. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.
In file: REX_REX.sol:620

Code

```
if (currentRxDay() >= 2) {                    // do nothing on days 0 and 1 where nothing happens
        if (currentRxDay() > globals.currentRxDay + 1) {        // snapshot needed? (difference between actual and update day > 2)
            _dailySnapshotPoint(globals.currentRxDay + 1); } }
```

Initial State:

Account: [CREATOR], balance: 0x801007, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [SOMEGUY], function: manualSnapshotOneDay(), txdata: 0x14229943, value: 0x0

Dependence on tx.origin
- SWC ID: 115
- Severity: Low
- Contract: RexToken
- Function name: `transferStake(bytes16,address)`
- PC address: 17299
- Estimated Gas Usage: 1119 - 1214

Description

Use of tx.origin as a part of authorization control.

The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.

In file: REX_REX.sol:1165

Code

```
require(_notContract(msg.sender) && msg.sender == tx.origin, 'REX: Not an address')
```

Initial State:

Account: [CREATOR], balance: 0x40000000001, nonce:0, storage:{}

Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0

Caller: [CREATOR], function: transferStake(bytes16,address), txdata: 0x88bcc0db00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000, value: 0x0

Dependence on tx.origin

- SWC ID: 115
- Severity: Low
- Contract: RexToken
- Function name: `initRexContracts(address,address,address,address)`
- PC address: 25519
- Estimated Gas Usage: 1315 - 1740

Description

Use of tx.origin as a part of authorization control.

The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.

In file: REX_REX.sol:1456

Code

```
require(msg.sender == TOKEN_DEFINER, 'RX: 28')
```

Initial State:

Account: [CREATOR], balance: 0x100000000823002, nonce:0, storage:{}

Account: [ATTACKER], balance: 0x1, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0

Caller: [ATTACKER], function: initRexContracts(address,address,address,address), txdata: 0x9ce26503000000000000000000000000deadbeefdeadbeefdeadbeefdeadbeefdeadbeef000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000, value: 0x0

Dependence on predictable environment variable

- SWC ID: 116
- Severity: Low
- Contract: RexToken
- Function name: `currentRxDay()`
- PC address: 26333
- Estimated Gas Usage: 1108 - 1203

Description

A control flow decision is made based on The block.timestamp environment variable.

The block.timestamp environment variable is used to determine a control flow decision. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

In file: REX_REX.sol:507

Code

```
block.timestamp >= LAUNCH_TIME ? _currentRxDay() : 0
```

Initial State:

Account: [CREATOR], balance: 0xa000044100000101, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [SOMEGUY], function: currentRxDay(), txdata: 0xa10e420e, value: 0x0

Dependence on tx.origin
- SWC ID: 115
- Severity: Low
- Contract: RexToken
- Function name: `mintSupply(address,uint256)`
- PC address: 34736
- Estimated Gas Usage: 1222 - 1647

Description

Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
In file: REX_REX.sol:1483

Code

```
msg.sender == AIRDROP_CONTRACT || msg.sender == RDA_CONTRACT
```

Initial State:

Account: [CREATOR], balance: 0x2a012, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x1, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0

Caller: [ATTACKER], function: mintSupply(address,uint256), txdata: 0xe742806a00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000, value: 0x0

Dependence on tx.origin
- SWC ID: 115
- Severity: Low
- Contract: RexToken
- Function name: `mintSupply(address,uint256)`
- PC address: 34823
- Estimated Gas Usage: 2094 - 2849

Description

Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
In file: REX_REX.sol:1483

Code

```
require( (msg.sender == AIRDROP_CONTRACT || msg.sender == RDA_CONTRACT), 'RX: 30')
```

Initial State:

Account: [CREATOR], balance: 0x4000001, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: mintSupply(address,uint256), txdata: 0xe742806a00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000, value: 0x0

Dependence on tx.origin
- SWC ID: 115
- Severity: Low
- Contract: RexToken

- Function name: `approve(address,uint256)`
- PC address: 35045
- Estimated Gas Usage: 429 - 524

Description

Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
In file: REX_REX.sol:341

Code

```
require(owner != address(0), "BEP20: approve from 0x0")
```

Initial State:

Account: [CREATOR], balance: 0x122000822004, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [SOMEGUY], function: approve(address,uint256), txdata: 0x095ea7b3000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000, value: 0x0

Dependence on predictable environment variable
- SWC ID: 116
- Severity: Low
- Contract: RexToken
- Function name: `endStake(bytes16)`
- PC address: 35635
- Estimated Gas Usage: 2922 - 3347

Description

A control flow decision is made based on The block.timestamp environment variable.

The block.timestamp environment variable is used to determine a control flow decision. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.
In file: REX_REX.sol:639

Code

```
for (uint32 _day = globals.currentRxDay; _day < _updateDay; _day++) {

        if (_day >= 2)
        {
            scheduledToEndToday = scheduledToEnd[_day] + snapshots[_day - 1].scheduledToEnd;
            SnapShot memory snapshot = snapshots[_day];
            snapshot.scheduledToEnd = scheduledToEndToday;

            snapshot.totalShares =
                globals.totalShares > scheduledToEndToday ?
                globals.totalShares - scheduledToEndToday : 0;

             // as on day 0 and 1 _inflationAmount() might be zero, those days are skipped
            snapshot.inflationAmount =  snapshot.totalShares
                .mul(PRECISION_RATE)
                .div(
                    _inflationAmount(
                        totalStakedToday,
                        totalSupply(),
                        totalPenalties[_day]
                    )
                );

            snapshots[_day] = snapshot;

        }

        globals.currentRxDay++;

    }
```

Initial State:

Account: [CREATOR], balance: 0x8080008002b082, nonce:0, storage:{}

Account: [ATTACKER], balance: 0x8, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0

Caller: [ATTACKER], function: endStake(bytes16), txdata:
0x99c2def4000000000000000000000000deadbeefdeadbeefdeadbeefdeadbeefdeadbeef, value: 0x0

Dependence on tx.origin

- SWC ID: 115
- Severity: Low
- Contract: RexToken
- Function name: `transfer(address,uint256)`
- PC address: 37207
- Estimated Gas Usage: 449 - 544

Description

Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
In file: REX_REX.sol:319

Code

```
require(sender != address(0), "BEP20: tx from 0x0")
```

Initial State:

Account: [CREATOR], balance: 0x32000, nonce:0, storage:{}

Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0

Caller: [CREATOR], function: transfer(address,uint256), txdata: 0xa9059cbb00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000, value: 0x0

RDA Contract:

Dependence on tx.origin

- SWC ID: 115
- Severity: Low
- Contract: RexDailyAuction
- Function name: `initContracts(address,address)`
- PC address: 6305
- Estimated Gas Usage: 1248 - 1673

Description

Use of tx.origin as a part of authorization control.

The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.

In file: REX_RDA.sol:309

Code

```
require(
        msg.sender == TOKEN_DEFINER,
        'REX: Not allowed.'
    )
```

Initial State:

Account: [CREATOR], balance: 0x2100000000000001, nonce:0, storage:{}

Account: [ATTACKER], balance: 0x8000000, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0

Caller: [ATTACKER], function: initContracts(address,address), txdata: 0x02f19ecb01000200000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000, value: 0x0

Dependence on tx.origin
- SWC ID: 115
- Severity: Low
- Contract: RexDailyAuction
- Function name: `donateBUSD(uint256,address)`
- PC address: 6943
- Estimated Gas Usage: 1133 - 1228

Description

Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
In file: REX_RDA.sol:299

Code

```
require(_notContract(msg.sender) && msg.sender == tx.origin, 'REX: Invalid sender.')
```

Initial State:

Account: [CREATOR], balance: 0x2000000000002001, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [SOMEGUY], function: donateBUSD(uint256,address), txdata: 0x03a8fff3000101020800000401000001000000000000000000000000000000000000000000000100000000000000000000000000000000000000000000000000000000000000001100108, value: 0x0

Dependence on tx.origin
- SWC ID: 115
- Severity: Low
- Contract: RexDailyAuction
- Function name: `claimStakeFromDonations(uint32)`
- PC address: 9114
- Estimated Gas Usage: 1108 - 1203

Description

Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
In file: REX_RDA.sol:299

Code

```
require(_notContract(msg.sender) && msg.sender == tx.origin, 'REX: Invalid sender.')
```

Initial State:

Account: [CREATOR], balance: 0x420200120032081, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x10, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [ATTACKER], function: claimStakeFromDonations(uint32), txdata: 0x1a350d6100000000000000000000800000000000000000000000000000000000000000000000, value: 0x0

Dependence on tx.origin
- SWC ID: 115
- Severity: Low
- Contract: RexDailyAuction
- Function name: `claimBusdFromTREASURY()`
- PC address: 12879
- Estimated Gas Usage: 1041 - 1136

Description

Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
In file: REX_RDA.sol:299

Code

```
require(_notContract(msg.sender) && msg.sender == tx.origin, 'REX: Invalid sender.')
```

Initial State:

Account: [CREATOR], balance: 0x88044000000009, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x80, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [ATTACKER], function: claimBusdFromTREASURY(), txdata: 0x5bd04aec, value: 0x0

Dependence on tx.origin
- SWC ID: 115
- Severity: Low
- Contract: RexDailyAuction
- Function name: `claimBusdFromReferrals()`
- PC address: 15723
- Estimated Gas Usage: 1085 - 1180

Description

Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
In file: REX_RDA.sol:299

Code

```
require(_notContract(msg.sender) && msg.sender == tx.origin, 'REX: Invalid sender.')
```

Initial State:

Account: [CREATOR], balance: 0x1092510f91591403a, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x1, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [ATTACKER], function: claimBusdFromReferrals(), txdata: 0x64c94a86, value: 0x0

Dependence on tx.origin
- SWC ID: 115
- Severity: Low
- Contract: RexDailyAuction
- Function name: `claimRexFromReferrals()`
- PC address: 16638
- Estimated Gas Usage: 1107 - 1202

Description

Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
In file: REX_RDA.sol:299

Code

```
require(_notContract(msg.sender) && msg.sender == tx.origin, 'REX: Invalid sender.')
```

Initial State:

Account: [CREATOR], balance: 0x100000000000001, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: claimRexFromReferrals(), txdata: 0x67302ce4, value: 0x0

Dependence on tx.origin
- SWC ID: 115
- Severity: Low
- Contract: RexDailyAuction

- Function name: `claimBusdFromBPD()`
- PC address: 18187
- Estimated Gas Usage: 1064 - 1159

Description

Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
In file: REX_RDA.sol:299

Code

```
require(_notContract(msg.sender) && msg.sender == tx.origin, 'REX: Invalid sender.')
```

Initial State:

Account: [CREATOR], balance: 0x80000001379a09, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x1, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [ATTACKER], function: claimBusdFromBPD(), txdata: 0x86c17b24, value: 0x0

Dependence on tx.origin
- SWC ID: 115
- Severity: Low
- Contract: RexDailyAuction
- Function name: `claimRexFromDonations()`
- PC address: 19511
- Estimated Gas Usage: 1085 - 1180

Description

Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.

In file: REX_RDA.sol:299

Code

```
require(_notContract(msg.sender) && msg.sender == tx.origin, 'REX: Invalid sender.')
```

Initial State:

Account: [CREATOR], balance: 0x5315b001001053501, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x1, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0
Caller: [ATTACKER], function: claimRexFromDonations(), txdata: 0xbad5db28, value: 0x0

Dependence on tx.origin
- SWC ID: 115
- Severity: Low
- Contract: RexDailyAuction
- Function name: `triggerDailyRoutineOneDay()`
- PC address: 21022
- Estimated Gas Usage: 1104 - 1199

Description

Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
In file: REX_RDA.sol:559

Code

```
require(_notContract(msg.sender) && msg.sender == tx.origin, 'REX: No contracts')
```

Initial State:

Account: [CREATOR], balance: 0x41181040550000001, nonce:0, storage:{}

Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0

Caller: [SOMEGUY], function: triggerDailyRoutineOneDay(), txdata: 0xbae91fab, value: 0x0

Dependence on tx.origin

- SWC ID: 115
- Severity: Low
- Contract: RexDailyAuction
- Function name: `withdrawLPTokens()`
- PC address: 22241
- Estimated Gas Usage: 1040 - 1135

Description

Use of tx.origin as a part of authorization control.

The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.

In file: REX_RDA.sol:299

Code

```
require(_notContract(msg.sender) && msg.sender == tx.origin, 'REX: Invalid sender.')
```

Initial State:

Account: [CREATOR], balance: 0x2, nonce:0, storage:{}

Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}

Transaction Sequence

Caller: [CREATOR], calldata: , value: 0x0

Caller: [SOMEGUY], function: withdrawLPTokens(), txdata: 0xedf8bc13, value: 0x0