0x Guard

Smart contracts security assessment

Final report

Tariff: Standard

The Money Tree

August 2023

0xguard.com

hello@0xguard.com

Contents

Introduction

The report has been prepared for **The Money Tree**.

The Money Tree project is a lottery type staking with referral program. It allows user to deposit ERC-20 token of a specific amount to become a member of one of 4 groups with different parameters for entering and possible outcome.

The code in the @theMoneyTreeDefi/theMoneyTree-contracts Github repo was audited in the 1d2ed23 commit.

The updated code was rechecked after the commit ce7e2db and deployed to 0xEaE382adf90e28603b9D9f49E4207bc5051370c9 in the BNB Smart Chain.

Name	The Money Tree
Audit date	2023-08-24 - 2023-08-25
Language	Solidity
Platform	Binance Smart Chain

Contracts checked

Name	Address
MoneyTree	0xEaE382adf90e28603b9D9f49E4207bc5051370c9

Procedure

We perform our audit according to the following procedure:

Automated analysis

- Scanning the project's smart contracts with several publicly available automated Solidity analysis tools

- Manual verification (reject or confirm) all the issues found by the tools

Manual audit

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

Known vulnerabilities checked

Title	Check result
Unencrypted Private Data On-Chain	passed
Code With No Effects	passed
Message call with hardcoded gas amount	passed
Typographical Error	passed
DoS With Block Gas Limit	not passed
Presence of unused variables	passed
Incorrect Inheritance Order	passed
Requirement Violation	passed
Weak Sources of Randomness from Chain Attributes	passed
Shadowing State Variables	passed
Incorrect Constructor Name	passed
Block values as a proxy for time	passed
Authorization through tx.origin	passed
DoS with Failed Call	passed
Delegatecall to Untrusted Callee	passed
Use of Deprecated Solidity Functions	passed

Assert Violation	passed
State Variable Default Visibility	passed
Reentrancy	passed
Unprotected SELFDESTRUCT Instruction	passed
Unprotected Ether Withdrawal	passed
Unchecked Call Return Value	passed
Floating Pragma	passed
Outdated Compiler Version	passed
Integer Overflow and Underflow	passed
Function Default Visibility	passed

Classification of issue severity

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

Issues

High severity issues

1. Owner controls the randomness (MoneyTree)

Status: Fixed

The winners are decided by a random index in the array of users. The randomness is defined with a combination of on-chain parameters and seed from Chainlink VRF oracle. But it's a `keeper` address (managed by the owner) who requests the seed from Oracle and who chooses between provided seeds, i.e., randomness `requestId` is not fixed for a specific round of lottery. Moreover, different distribution round within the same epoch may be called with different random seeds (`requestId`).

```
    function requestRandomWords() external onlyKeeper returns (uint256 requestId) {
        requestId = requestRandomness(
            callbackGasLimit,
            requestConfirmations,
            numWords
        );
        s_requests[requestId] = RequestStatus({
            paid: VRF_V2_WRAPPER.calculateRequestPrice(callbackGasLimit),
            randomWords: new uint256[](0),
            fulfilled: false
        });
        requestIds.push(requestId);
        lastRequestId = requestId;
        emit RequestSent(requestId, numWords);
        return requestId;
    }

    function processRandomness(uint256 _requestId, uint256 _k, uint256 _size) private
returns (uint256 _randomness) {
        (,,uint256[] memory _randomWords) = getRequestStatus(_requestId);
        nonce++;
        _randomness = uint256(keccak256(abi.encode(_randomWords[_k],
blockhash(block.number), _size, nonce)));
        _randomness = _randomness % _size;
    }

    function distrubuteStep01(uint256 _requestId) external onlyKeeper returns (bool) {
        (, bool fulfilled,) = getRequestStatus(_requestId);
    ...
    }

    function distrubuteStep02(uint256 _requestId) external onlyKeeper returns (bool) {
        v.winnerIndex = processRandomness(_requestId, 0, v.len);
    ...
    }
```

Recommendation: Fix `requestId` for `epoch`. It may be also useful to mandatory check that distrubuteStep02 and next steps are called not in the same block as the distrubuteStep01. That increases the randomness (which includes the `blockhash` of previous block).

Medium severity issues

1. Possible locked funds (MoneyTree)

Status: Open

The funds distribution is managed by the owner through the `setGroupsInfo` function, which requires the sum of distribution percents to be equal to 100%. However, input elements in the _groups array may be duplicated, thus the resulting total distributed percent may be less than 100%.

```
    function setGroupsInfo(Group[] memory _groups, GroupInfo[] memory _infos) external
onlyOwner returns (bool) {
        if (_groups.length != _infos.length) revert MoneyTreeInvalidGroupsParameters();
        uint256 sum;
        for (uint256 i = 0; i < _groups.length; i++) {
            groupInfo[_groups[i]].depositSize = _infos[i].depositSize;
            groupInfo[_groups[i]].maxPayout = _infos[i].maxPayout;
            groupInfo[_groups[i]].distributionPercent = _infos[i].distributionPercent;
            sum += _infos[i].distributionPercent;
        }
        if (sum != DIVIDER) revert MoneyTreeInvalidGroupsParameters();
        return true;
    }
```

The second option for locking is that keeper may miss the distribution window within the current epoch.

```
    function distrubuteStep01(uint256 _requestId) external onlyKeeper returns (bool) {
        uint256 currentEpoch = getEpoch(block.timestamp);
        if (isTimeInWindow(block.timestamp)) revert MoneyTreeWindowwIsOpen();
    ...
    }
```

Recommendation: Include an explicit check over the updated values to be summed up to 100%.

Include epoch parameter into the distribution functions.

Update: In the updated code the `setGroupsInfo` function can be used only as initializer. We recommend explicitly justifying the correctness of this behavior.

the second locking scenario is still available.

2. Gas block limit (MoneyTree)

Status: Open

The unlimited number of users may cause one of distribution steps starting from distrubuteStep02 to constantly fail if for loop iterating over the group member list may exceed the block gas limit.

Recommendation: Estimate gas consumption and limit the number of the user according to selected network and its block gas limit.

3. Wrong usage of blockhash function (MoneyTree)

Status: Fixed

The random index calculation includes the hash of the current block which is always 0. the `blockhash` function may be used to obtain hashes for 256 recent blocks, the `block.number` is not included.

```
    function processRandomness(uint256 _requestId, uint256 _k, uint256 _size) private
returns (uint256 _randomness) {
        (,,uint256[] memory _randomWords) = getRequestStatus(_requestId);
        nonce++;
        _randomness = uint256(keccak256(abi.encode(_randomWords[_k],
blockhash(block.number), _size, nonce)));
        _randomness = _randomness % _size;
    }
```

Recommendation: Use `blockhash(block.number - 1)`.

Low severity issues

1. Import error (MoneyTree)
Status: Fixed

The contract has wrong import of the VRFV2WrapperConsumerBase.

2. Typographical errors (MoneyTree)
Status: Open

Typos found in `recieved`, `distrubute`, `Windoww`.

3. Variables with default visibility (MoneyTree)
Status: Fixed

No visibility is defined for the `callbackGasLimit`, `requestConfirmations`, `numWords` variables.

4. Missing event (MoneyTree)
Status: Fixed

The dev distribution in the distrubuteStep01 function iterates the _dev list and transfers them equal shares. The last address is treated different and the `DevBonusPaid` event is not emitted.

5. Unsafe initialization (MoneyTree)
Status: Fixed

The dev group length is checked to be exactly 11 in length during the initialization. However, actual group member count may be less as the result of `EnumerableSet.add` function is not checked - duplicated addresses will be ignored.

```
    function initialize(address[] memory _devs, address _tradingAccount) external
initializer onlyOwner returns (bool) {
        uint256 len = _devs.length;
        if (len != 11) revert MoneyTreeInvalidDevsLength(len);
        if (_tradingAccount == address(0)) revert MoneyTreeInvalidAddress(address(0));

        for (uint256 i = 0; i < len; i++) {
            if (_devs[i] == address(0)) revert MoneyTreeInvalidAddress(address(0));
            _dev.add(_devs[i]);
        return true;
    }

    function _add(Set storage set, bytes32 value) private returns (bool) {
        if (!_contains(set, value)) {
            set._values.push(value);
            // The value is stored at length-1, but we add 1 to all indexes
            // and use 0 as a sentinel value
            set._indexes[value] = set._values.length;
            return true;
        } else {
            return false;
        }
    }
```

Recommendation: Require only successful result of `add` function.

Conclusion

The Money Tree MoneyTree contract was audited. 1 high, 3 medium, 5 low severity issues were found.

1 high, 1 medium, 4 low severity issues have been fixed in the update.

Disclaimer

Slither output

```
INFO:Detectors:
MoneyTree.processRandomness(uint256,uint256,uint256) (contracts/
themoneytree.sol#1028-1033) uses a weak PRNG: "_randomness = _randomness % _size
(contracts/themoneytree.sol#1032)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG
INFO:Detectors:
MoneyTree.withdrawLink() (contracts/themoneytree.sol#803-807) ignores return value by
link.transfer(msg.sender,link.balanceOf(address(this))) (contracts/
themoneytree.sol#805)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-
transfer
INFO:Detectors:
MoneyTree.isTimeInWindow(uint256) (contracts/themoneytree.sol#911-915) performs a
multiplication on the result of a division:
	- diff - (diff / 604800) * 604800 < 86400 (contracts/themoneytree.sol#914)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-
multiply
INFO:Detectors:
MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/themoneytree.sol#212-294)
uses a dangerous strict equality:
	- userInfo[_referrer].lastEpochAddReferrals == currentEpoch (contracts/
themoneytree.sol#240)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-
strict-equalities
INFO:Detectors:
Reentrancy in MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/
themoneytree.sol#212-294):
	External calls:
	- IERC20(token).safeTransferFrom(_sender,address(this),_amount) (contracts/
themoneytree.sol#225)
	State variables written after the call(s):
	- userInfo[_sender].group = _group (contracts/themoneytree.sol#234)
	MoneyTree.userInfo (contracts/themoneytree.sol#105) can be used in cross function
reentrancies:
	- MoneyTree.addUserToGroupCurrentEpochList(address) (contracts/
themoneytree.sol#988-995)
	- MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/
themoneytree.sol#212-294)
```

```
- MoneyTree.distrubuteStep02(uint256) (contracts/themoneytree.sol#368-468)
- MoneyTree.distrubuteStep03(uint256) (contracts/themoneytree.sol#471-572)
- MoneyTree.distrubuteStep04(uint256) (contracts/themoneytree.sol#575-675)
- MoneyTree.distrubuteStep05(uint256) (contracts/themoneytree.sol#678-743)
- MoneyTree.getUserInfo(address) (contracts/themoneytree.sol#767-790)
- MoneyTree.payWinner(address) (contracts/themoneytree.sol#944-985)
- MoneyTree.removeUserFromGroupCurrentEpochList(address) (contracts/
themoneytree.sol#998-1014)
- MoneyTree.userInfo (contracts/themoneytree.sol#105)
- userInfo[_sender].deposited = true (contracts/themoneytree.sol#235)
MoneyTree.userInfo (contracts/themoneytree.sol#105) can be used in cross function
reentrancies:
- MoneyTree.addUserToGroupCurrentEpochList(address) (contracts/
themoneytree.sol#988-995)
- MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/
themoneytree.sol#212-294)
- MoneyTree.distrubuteStep02(uint256) (contracts/themoneytree.sol#368-468)
- MoneyTree.distrubuteStep03(uint256) (contracts/themoneytree.sol#471-572)
- MoneyTree.distrubuteStep04(uint256) (contracts/themoneytree.sol#575-675)
- MoneyTree.distrubuteStep05(uint256) (contracts/themoneytree.sol#678-743)
- MoneyTree.getUserInfo(address) (contracts/themoneytree.sol#767-790)
- MoneyTree.payWinner(address) (contracts/themoneytree.sol#944-985)
- MoneyTree.removeUserFromGroupCurrentEpochList(address) (contracts/
themoneytree.sol#998-1014)
- MoneyTree.userInfo (contracts/themoneytree.sol#105)
- userInfo[_referrer].numberOfReferrals ++ (contracts/themoneytree.sol#241)
MoneyTree.userInfo (contracts/themoneytree.sol#105) can be used in cross function
reentrancies:
- MoneyTree.addUserToGroupCurrentEpochList(address) (contracts/
themoneytree.sol#988-995)
- MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/
themoneytree.sol#212-294)
- MoneyTree.distrubuteStep02(uint256) (contracts/themoneytree.sol#368-468)
- MoneyTree.distrubuteStep03(uint256) (contracts/themoneytree.sol#471-572)
- MoneyTree.distrubuteStep04(uint256) (contracts/themoneytree.sol#575-675)
- MoneyTree.distrubuteStep05(uint256) (contracts/themoneytree.sol#678-743)
- MoneyTree.getUserInfo(address) (contracts/themoneytree.sol#767-790)
- MoneyTree.payWinner(address) (contracts/themoneytree.sol#944-985)
- MoneyTree.removeUserFromGroupCurrentEpochList(address) (contracts/
themoneytree.sol#998-1014)
- MoneyTree.userInfo (contracts/themoneytree.sol#105)
- userInfo[_referrer].numberOfReferrals = 1 (contracts/themoneytree.sol#243)
MoneyTree.userInfo (contracts/themoneytree.sol#105) can be used in cross function
reentrancies:
- MoneyTree.addUserToGroupCurrentEpochList(address) (contracts/
themoneytree.sol#988-995)
- MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/
themoneytree.sol#212-294)
- MoneyTree.distrubuteStep02(uint256) (contracts/themoneytree.sol#368-468)
- MoneyTree.distrubuteStep03(uint256) (contracts/themoneytree.sol#471-572)
- MoneyTree.distrubuteStep04(uint256) (contracts/themoneytree.sol#575-675)
- MoneyTree.distrubuteStep05(uint256) (contracts/themoneytree.sol#678-743)
- MoneyTree.getUserInfo(address) (contracts/themoneytree.sol#767-790)
- MoneyTree.payWinner(address) (contracts/themoneytree.sol#944-985)
- MoneyTree.removeUserFromGroupCurrentEpochList(address) (contracts/
themoneytree.sol#998-1014)
- MoneyTree.userInfo (contracts/themoneytree.sol#105)
- userInfo[_referrer].lastEpochAddReferrals = currentEpoch (contracts/
themoneytree.sol#245)
MoneyTree.userInfo (contracts/themoneytree.sol#105) can be used in cross function
reentrancies:
- MoneyTree.addUserToGroupCurrentEpochList(address) (contracts/
themoneytree.sol#988-995)
- MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/
themoneytree.sol#212-294)
- MoneyTree.distrubuteStep02(uint256) (contracts/themoneytree.sol#368-468)
- MoneyTree.distrubuteStep03(uint256) (contracts/themoneytree.sol#471-572)
- MoneyTree.distrubuteStep04(uint256) (contracts/themoneytree.sol#575-675)
- MoneyTree.distrubuteStep05(uint256) (contracts/themoneytree.sol#678-743)
- MoneyTree.getUserInfo(address) (contracts/themoneytree.sol#767-790)
- MoneyTree.payWinner(address) (contracts/themoneytree.sol#944-985)
- MoneyTree.removeUserFromGroupCurrentEpochList(address) (contracts/
themoneytree.sol#998-1014)
- MoneyTree.userInfo (contracts/themoneytree.sol#105)
- userInfo[_referrer].deposited = false (contracts/themoneytree.sol#263)
MoneyTree.userInfo (contracts/themoneytree.sol#105) can be used in cross function
reentrancies:
- MoneyTree.addUserToGroupCurrentEpochList(address) (contracts/
themoneytree.sol#988-995)
- MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/
themoneytree.sol#212-294)
- MoneyTree.distrubuteStep02(uint256) (contracts/themoneytree.sol#368-468)
- MoneyTree.distrubuteStep03(uint256) (contracts/themoneytree.sol#471-572)
- MoneyTree.distrubuteStep04(uint256) (contracts/themoneytree.sol#575-675)
- MoneyTree.distrubuteStep05(uint256) (contracts/themoneytree.sol#678-743)
- MoneyTree.getUserInfo(address) (contracts/themoneytree.sol#767-790)
- MoneyTree.payWinner(address) (contracts/themoneytree.sol#944-985)
- MoneyTree.removeUserFromGroupCurrentEpochList(address) (contracts/
themoneytree.sol#998-1014)
- MoneyTree.userInfo (contracts/themoneytree.sol#105)
- userInfo[_referrer].totalReceived = 0 (contracts/themoneytree.sol#264)
MoneyTree.userInfo (contracts/themoneytree.sol#105) can be used in cross function
reentrancies:
- MoneyTree.addUserToGroupCurrentEpochList(address) (contracts/
themoneytree.sol#988-995)
- MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/
themoneytree.sol#212-294)
- MoneyTree.distrubuteStep02(uint256) (contracts/themoneytree.sol#368-468)
- MoneyTree.distrubuteStep03(uint256) (contracts/themoneytree.sol#471-572)
- MoneyTree.distrubuteStep04(uint256) (contracts/themoneytree.sol#575-675)
- MoneyTree.distrubuteStep05(uint256) (contracts/themoneytree.sol#678-743)
- MoneyTree.getUserInfo(address) (contracts/themoneytree.sol#767-790)
- MoneyTree.payWinner(address) (contracts/themoneytree.sol#944-985)
- MoneyTree.removeUserFromGroupCurrentEpochList(address) (contracts/
themoneytree.sol#998-1014)
- MoneyTree.userInfo (contracts/themoneytree.sol#105)
- userInfo[_referrer].numberOfReferrals = 0 (contracts/themoneytree.sol#265)
MoneyTree.userInfo (contracts/themoneytree.sol#105) can be used in cross function
reentrancies:
- MoneyTree.addUserToGroupCurrentEpochList(address) (contracts/
themoneytree.sol#988-995)
- MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/
themoneytree.sol#212-294)
- MoneyTree.distrubuteStep02(uint256) (contracts/themoneytree.sol#368-468)
- MoneyTree.distrubuteStep03(uint256) (contracts/themoneytree.sol#471-572)
- MoneyTree.distrubuteStep04(uint256) (contracts/themoneytree.sol#575-675)
- MoneyTree.distrubuteStep05(uint256) (contracts/themoneytree.sol#678-743)
- MoneyTree.getUserInfo(address) (contracts/themoneytree.sol#767-790)
- MoneyTree.payWinner(address) (contracts/themoneytree.sol#944-985)
- MoneyTree.removeUserFromGroupCurrentEpochList(address) (contracts/
themoneytree.sol#998-1014)
- MoneyTree.userInfo (contracts/themoneytree.sol#105)
- userInfo[_referrer].lastEpochAddReferrals = 0 (contracts/themoneytree.sol#266)
```

```
⊠MoneyTree.userInfo (contracts/themoneytree.sol#105) can be used in cross function
reentrancies:
⊠- MoneyTree.addUserToGroupCurrentEpochList(address) (contracts/
themoneytree.sol#988-995)
⊠- MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/
themoneytree.sol#212-294)
⊠- MoneyTree.distrubuteStep02(uint256) (contracts/themoneytree.sol#368-468)
⊠- MoneyTree.distrubuteStep03(uint256) (contracts/themoneytree.sol#471-572)
⊠- MoneyTree.distrubuteStep04(uint256) (contracts/themoneytree.sol#575-675)
⊠- MoneyTree.distrubuteStep05(uint256) (contracts/themoneytree.sol#678-743)
⊠- MoneyTree.getUserInfo(address) (contracts/themoneytree.sol#767-790)
⊠- MoneyTree.payWinner(address) (contracts/themoneytree.sol#944-985)
⊠- MoneyTree.removeUserFromGroupCurrentEpochList(address) (contracts/
themoneytree.sol#998-1014)
⊠- MoneyTree.userInfo (contracts/themoneytree.sol#105)
⊠- userInfo[_referrer].winner = false (contracts/themoneytree.sol#267)
⊠MoneyTree.userInfo (contracts/themoneytree.sol#105) can be used in cross function
reentrancies:
⊠- MoneyTree.addUserToGroupCurrentEpochList(address) (contracts/
themoneytree.sol#988-995)
⊠- MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/
themoneytree.sol#212-294)
⊠- MoneyTree.distrubuteStep02(uint256) (contracts/themoneytree.sol#368-468)
⊠- MoneyTree.distrubuteStep03(uint256) (contracts/themoneytree.sol#471-572)
⊠- MoneyTree.distrubuteStep04(uint256) (contracts/themoneytree.sol#575-675)
⊠- MoneyTree.distrubuteStep05(uint256) (contracts/themoneytree.sol#678-743)
⊠- MoneyTree.getUserInfo(address) (contracts/themoneytree.sol#767-790)
⊠- MoneyTree.payWinner(address) (contracts/themoneytree.sol#944-985)
⊠- MoneyTree.removeUserFromGroupCurrentEpochList(address) (contracts/
themoneytree.sol#998-1014)
⊠- MoneyTree.userInfo (contracts/themoneytree.sol#105)
⊠- userInfo[_referrer].totalReceived += depositSize * 2 (contracts/
themoneytree.sol#280)
⊠MoneyTree.userInfo (contracts/themoneytree.sol#105) can be used in cross function
reentrancies:
⊠- MoneyTree.addUserToGroupCurrentEpochList(address) (contracts/
themoneytree.sol#988-995)
⊠- MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/
themoneytree.sol#212-294)
⊠- MoneyTree.distrubuteStep02(uint256) (contracts/themoneytree.sol#368-468)
⊠- MoneyTree.distrubuteStep03(uint256) (contracts/themoneytree.sol#471-572)
```

```
- MoneyTree.distrubuteStep04(uint256) (contracts/themoneytree.sol#575-675)
- MoneyTree.distrubuteStep05(uint256) (contracts/themoneytree.sol#678-743)
- MoneyTree.getUserInfo(address) (contracts/themoneytree.sol#767-790)
- MoneyTree.payWinner(address) (contracts/themoneytree.sol#944-985)
- MoneyTree.removeUserFromGroupCurrentEpochList(address) (contracts/
themoneytree.sol#998-1014)
- MoneyTree.userInfo (contracts/themoneytree.sol#105)
- userInfo[_referrer].numberOfReferrals = 0 (contracts/themoneytree.sol#281)
MoneyTree.userInfo (contracts/themoneytree.sol#105) can be used in cross function
reentrancies:
- MoneyTree.addUserToGroupCurrentEpochList(address) (contracts/
themoneytree.sol#988-995)
- MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/
themoneytree.sol#212-294)
- MoneyTree.distrubuteStep02(uint256) (contracts/themoneytree.sol#368-468)
- MoneyTree.distrubuteStep03(uint256) (contracts/themoneytree.sol#471-572)
- MoneyTree.distrubuteStep04(uint256) (contracts/themoneytree.sol#575-675)
- MoneyTree.distrubuteStep05(uint256) (contracts/themoneytree.sol#678-743)
- MoneyTree.getUserInfo(address) (contracts/themoneytree.sol#767-790)
- MoneyTree.payWinner(address) (contracts/themoneytree.sol#944-985)
- MoneyTree.removeUserFromGroupCurrentEpochList(address) (contracts/
themoneytree.sol#998-1014)
- MoneyTree.userInfo (contracts/themoneytree.sol#105)
Reentrancy in MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/
themoneytree.sol#212-294):
External calls:
- IERC20(token).safeTransferFrom(_sender,address(this),_amount) (contracts/
themoneytree.sol#225)
- IERC20(token).safeTransfer(_referrer,referrerMaxPayout - receivedAmount) (contracts/
themoneytree.sol#273)
State variables written after the call(s):
- epochDepositAmount[currentEpoch] -= (referrerMaxPayout - receivedAmount) (contracts/
themoneytree.sol#275)
MoneyTree.epochDepositAmount (contracts/themoneytree.sol#112) can be used in cross
function reentrancies:
- MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/
themoneytree.sol#212-294)
- MoneyTree.distrubuteStep01(uint256) (contracts/themoneytree.sol#321-365)
- MoneyTree.getEpochDepositAmount(uint256) (contracts/themoneytree.sol#798-800)
Reentrancy in MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/
themoneytree.sol#212-294):
```

```
External calls:
- IERC20(token).safeTransferFrom(_sender,address(this),_amount) (contracts/
themoneytree.sol#225)
- IERC20(token).safeTransfer(_referrer,depositSize * 2) (contracts/
themoneytree.sol#282)
State variables written after the call(s):
- epochDepositAmount[currentEpoch] -= depositSize * 2 (contracts/themoneytree.sol#284)
MoneyTree.epochDepositAmount (contracts/themoneytree.sol#112) can be used in cross
function reentrancies:
- MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/
themoneytree.sol#212-294)
- MoneyTree.distrubuteStep01(uint256) (contracts/themoneytree.sol#321-365)
- MoneyTree.getEpochDepositAmount(uint256) (contracts/themoneytree.sol#798-800)
Reentrancy in MoneyTree.distrubuteStep01(uint256) (contracts/themoneytree.sol#321-365):
External calls:
- IERC20(token).safeTransfer(v.recipient,v.distributeAmountDev - v.devPaymentAmount *
10) (contracts/themoneytree.sol#348)
- IERC20(token).safeTransfer(v.recipient,v.devPaymentAmount) (contracts/
themoneytree.sol#352)
- IERC20(token).safeTransfer(tradingAccount,v.distributeAmountTrading) (contracts/
themoneytree.sol#358)
State variables written after the call(s):
- epochStepDone[currentEpoch][1] = true (contracts/themoneytree.sol#360)
MoneyTree.epochStepDone (contracts/themoneytree.sol#117) can be used in cross function
reentrancies:
- MoneyTree.distrubuteStep01(uint256) (contracts/themoneytree.sol#321-365)
- MoneyTree.distrubuteStep02(uint256) (contracts/themoneytree.sol#368-468)
- MoneyTree.distrubuteStep03(uint256) (contracts/themoneytree.sol#471-572)
- MoneyTree.distrubuteStep04(uint256) (contracts/themoneytree.sol#575-675)
- MoneyTree.distrubuteStep05(uint256) (contracts/themoneytree.sol#678-743)
Reentrancy in MoneyTree.distrubuteStep05(uint256) (contracts/themoneytree.sol#678-743):
External calls:
- IERC20(token).safeTransfer(v.winnerAddress,v.winnerPayment) (contracts/
themoneytree.sol#710)
- IERC20(token).safeTransfer(v.winnerAddress,v.distributeAmountLottery) (contracts/
themoneytree.sol#729)
State variables written after the call(s):
- epochStepDone[currentEpoch][4] = true (contracts/themoneytree.sol#738)
MoneyTree.epochStepDone (contracts/themoneytree.sol#117) can be used in cross function
reentrancies:
- MoneyTree.distrubuteStep01(uint256) (contracts/themoneytree.sol#321-365)
```

```
⊠- MoneyTree.distrubuteStep02(uint256) (contracts/themoneytree.sol#368-468)
⊠- MoneyTree.distrubuteStep03(uint256) (contracts/themoneytree.sol#471-572)
⊠- MoneyTree.distrubuteStep04(uint256) (contracts/themoneytree.sol#575-675)
⊠- MoneyTree.distrubuteStep05(uint256) (contracts/themoneytree.sol#678-743)
⊠- v.winnerIndex = processRandomness(_requestId,3,v.len) (contracts/
themoneytree.sol#692)
⊠⊠- nonce ++ (contracts/themoneytree.sol#1030)
⊠MoneyTree.nonce (contracts/themoneytree.sol#81) can be used in cross function
reentrancies:
⊠- MoneyTree.processRandomness(uint256,uint256,uint256) (contracts/
themoneytree.sol#1028-1033)
⊠- userInfo[v.winnerAddress].deposited = false (contracts/themoneytree.sol#712)
⊠MoneyTree.userInfo (contracts/themoneytree.sol#105) can be used in cross function
reentrancies:
⊠- MoneyTree.addUserToGroupCurrentEpochList(address) (contracts/
themoneytree.sol#988-995)
⊠- MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/
themoneytree.sol#212-294)
⊠- MoneyTree.distrubuteStep02(uint256) (contracts/themoneytree.sol#368-468)
⊠- MoneyTree.distrubuteStep03(uint256) (contracts/themoneytree.sol#471-572)
⊠- MoneyTree.distrubuteStep04(uint256) (contracts/themoneytree.sol#575-675)
⊠- MoneyTree.distrubuteStep05(uint256) (contracts/themoneytree.sol#678-743)
⊠- MoneyTree.getUserInfo(address) (contracts/themoneytree.sol#767-790)
⊠- MoneyTree.payWinner(address) (contracts/themoneytree.sol#944-985)
⊠- MoneyTree.removeUserFromGroupCurrentEpochList(address) (contracts/
themoneytree.sol#998-1014)
⊠- MoneyTree.userInfo (contracts/themoneytree.sol#105)
⊠- userInfo[v.winnerAddress].totalReceived = 0 (contracts/themoneytree.sol#713)
⊠MoneyTree.userInfo (contracts/themoneytree.sol#105) can be used in cross function
reentrancies:
⊠- MoneyTree.addUserToGroupCurrentEpochList(address) (contracts/
themoneytree.sol#988-995)
⊠- MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/
themoneytree.sol#212-294)
⊠- MoneyTree.distrubuteStep02(uint256) (contracts/themoneytree.sol#368-468)
⊠- MoneyTree.distrubuteStep03(uint256) (contracts/themoneytree.sol#471-572)
⊠- MoneyTree.distrubuteStep04(uint256) (contracts/themoneytree.sol#575-675)
⊠- MoneyTree.distrubuteStep05(uint256) (contracts/themoneytree.sol#678-743)
⊠- MoneyTree.getUserInfo(address) (contracts/themoneytree.sol#767-790)
⊠- MoneyTree.payWinner(address) (contracts/themoneytree.sol#944-985)
⊠- MoneyTree.removeUserFromGroupCurrentEpochList(address) (contracts/
themoneytree.sol#998-1014)
```

```
⊠- MoneyTree.userInfo (contracts/themoneytree.sol#105)
⊠- userInfo[v.winnerAddress].numberOfReferrals = 0 (contracts/themoneytree.sol#714)
⊠MoneyTree.userInfo (contracts/themoneytree.sol#105) can be used in cross function
reentrancies:
⊠- MoneyTree.addUserToGroupCurrentEpochList(address) (contracts/
themoneytree.sol#988-995)
⊠- MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/
themoneytree.sol#212-294)
⊠- MoneyTree.distrubuteStep02(uint256) (contracts/themoneytree.sol#368-468)
⊠- MoneyTree.distrubuteStep03(uint256) (contracts/themoneytree.sol#471-572)
⊠- MoneyTree.distrubuteStep04(uint256) (contracts/themoneytree.sol#575-675)
⊠- MoneyTree.distrubuteStep05(uint256) (contracts/themoneytree.sol#678-743)
⊠- MoneyTree.getUserInfo(address) (contracts/themoneytree.sol#767-790)
⊠- MoneyTree.payWinner(address) (contracts/themoneytree.sol#944-985)
⊠- MoneyTree.removeUserFromGroupCurrentEpochList(address) (contracts/
themoneytree.sol#998-1014)
⊠- MoneyTree.userInfo (contracts/themoneytree.sol#105)
⊠- userInfo[v.winnerAddress].lastEpochAddReferrals = 0 (contracts/themoneytree.sol#715)
⊠MoneyTree.userInfo (contracts/themoneytree.sol#105) can be used in cross function
reentrancies:
⊠- MoneyTree.addUserToGroupCurrentEpochList(address) (contracts/
themoneytree.sol#988-995)
⊠- MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/
themoneytree.sol#212-294)
⊠- MoneyTree.distrubuteStep02(uint256) (contracts/themoneytree.sol#368-468)
⊠- MoneyTree.distrubuteStep03(uint256) (contracts/themoneytree.sol#471-572)
⊠- MoneyTree.distrubuteStep04(uint256) (contracts/themoneytree.sol#575-675)
⊠- MoneyTree.distrubuteStep05(uint256) (contracts/themoneytree.sol#678-743)
⊠- MoneyTree.getUserInfo(address) (contracts/themoneytree.sol#767-790)
⊠- MoneyTree.payWinner(address) (contracts/themoneytree.sol#944-985)
⊠- MoneyTree.removeUserFromGroupCurrentEpochList(address) (contracts/
themoneytree.sol#998-1014)
⊠- MoneyTree.userInfo (contracts/themoneytree.sol#105)
⊠- userInfo[v.winnerAddress].winner = false (contracts/themoneytree.sol#716)
⊠MoneyTree.userInfo (contracts/themoneytree.sol#105) can be used in cross function
reentrancies:
⊠- MoneyTree.addUserToGroupCurrentEpochList(address) (contracts/
themoneytree.sol#988-995)
⊠- MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/
themoneytree.sol#212-294)
⊠- MoneyTree.distrubuteStep02(uint256) (contracts/themoneytree.sol#368-468)
```

```
- MoneyTree.distrubuteStep03(uint256) (contracts/themoneytree.sol#471-572)
- MoneyTree.distrubuteStep04(uint256) (contracts/themoneytree.sol#575-675)
- MoneyTree.distrubuteStep05(uint256) (contracts/themoneytree.sol#678-743)
- MoneyTree.getUserInfo(address) (contracts/themoneytree.sol#767-790)
- MoneyTree.payWinner(address) (contracts/themoneytree.sol#944-985)
- MoneyTree.removeUserFromGroupCurrentEpochList(address) (contracts/
themoneytree.sol#998-1014)
- MoneyTree.userInfo (contracts/themoneytree.sol#105)
- userInfo[v.winnerAddress].totalReceived += v.distributeAmountLottery (contracts/
themoneytree.sol#727)
MoneyTree.userInfo (contracts/themoneytree.sol#105) can be used in cross function
reentrancies:
- MoneyTree.addUserToGroupCurrentEpochList(address) (contracts/
themoneytree.sol#988-995)
- MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/
themoneytree.sol#212-294)
- MoneyTree.distrubuteStep02(uint256) (contracts/themoneytree.sol#368-468)
- MoneyTree.distrubuteStep03(uint256) (contracts/themoneytree.sol#471-572)
- MoneyTree.distrubuteStep04(uint256) (contracts/themoneytree.sol#575-675)
- MoneyTree.distrubuteStep05(uint256) (contracts/themoneytree.sol#678-743)
- MoneyTree.getUserInfo(address) (contracts/themoneytree.sol#767-790)
- MoneyTree.payWinner(address) (contracts/themoneytree.sol#944-985)
- MoneyTree.removeUserFromGroupCurrentEpochList(address) (contracts/
themoneytree.sol#998-1014)
- MoneyTree.userInfo (contracts/themoneytree.sol#105)
- userInfo[v.winnerAddress].numberOfReferrals = 0 (contracts/themoneytree.sol#728)
MoneyTree.userInfo (contracts/themoneytree.sol#105) can be used in cross function
reentrancies:
- MoneyTree.addUserToGroupCurrentEpochList(address) (contracts/
themoneytree.sol#988-995)
- MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/
themoneytree.sol#212-294)
- MoneyTree.distrubuteStep02(uint256) (contracts/themoneytree.sol#368-468)
- MoneyTree.distrubuteStep03(uint256) (contracts/themoneytree.sol#471-572)
- MoneyTree.distrubuteStep04(uint256) (contracts/themoneytree.sol#575-675)
- MoneyTree.distrubuteStep05(uint256) (contracts/themoneytree.sol#678-743)
- MoneyTree.getUserInfo(address) (contracts/themoneytree.sol#767-790)
- MoneyTree.payWinner(address) (contracts/themoneytree.sol#944-985)
- MoneyTree.removeUserFromGroupCurrentEpochList(address) (contracts/
themoneytree.sol#998-1014)
- MoneyTree.userInfo (contracts/themoneytree.sol#105)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-1
INFO:Detectors:
MoneyTree.distrubuteStep02(uint256) (contracts/themoneytree.sol#368-468) contains a
tautology or contradiction:
⊠- i_scope_0 >= 0 (contracts/themoneytree.sol#410)
MoneyTree.distrubuteStep03(uint256) (contracts/themoneytree.sol#471-572) contains a
tautology or contradiction:
⊠- i_scope_0 >= 0 (contracts/themoneytree.sol#513)
MoneyTree.distrubuteStep04(uint256) (contracts/themoneytree.sol#575-675) contains a
tautology or contradiction:
⊠- i_scope_0 >= 0 (contracts/themoneytree.sol#617)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#tautology-or-
contradiction
INFO:Detectors:
MoneyTree.distrubuteStep02(uint256).v (contracts/themoneytree.sol#373) is a local
variable never initialized
MoneyTree.distrubuteStep03(uint256).v (contracts/themoneytree.sol#476) is a local
variable never initialized
MoneyTree.distrubuteStep01(uint256).v (contracts/themoneytree.sol#332) is a local
variable never initialized
MoneyTree.distrubuteStep05(uint256).v (contracts/themoneytree.sol#683) is a local
variable never initialized
MoneyTree.distrubuteStep04(uint256).v (contracts/themoneytree.sol#580) is a local
variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-
local-variables
INFO:Detectors:
MoneyTree.initialize(address[],address) (contracts/themoneytree.sol#171-184) ignores
return value by _dev.add(_devs[i]) (contracts/themoneytree.sol#178)
MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/themoneytree.sol#212-294)
ignores return value by _stakersTotal.add(_sender) (contracts/themoneytree.sol#228)
MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/themoneytree.sol#212-294)
ignores return value by _stakersPoolA.add(_sender) (contracts/themoneytree.sol#229)
MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/themoneytree.sol#212-294)
ignores return value by _stakersPoolB.add(_sender) (contracts/themoneytree.sol#230)
MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/themoneytree.sol#212-294)
ignores return value by _stakersPoolC.add(_sender) (contracts/themoneytree.sol#231)
MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/themoneytree.sol#212-294)
ignores return value by _stakersPool_A_B.add(_sender) (contracts/themoneytree.sol#232)
MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/themoneytree.sol#212-294)
```

```
ignores return value by _stakersTotal.remove(_referrer) (contracts/
themoneytree.sol#255)
MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/themoneytree.sol#212-294)
ignores return value by _stakersPoolA.remove(_referrer) (contracts/
themoneytree.sol#256)
MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/themoneytree.sol#212-294)
ignores return value by _stakersPoolB.remove(_referrer) (contracts/
themoneytree.sol#257)
MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/themoneytree.sol#212-294)
ignores return value by _stakersPoolC.remove(_referrer) (contracts/
themoneytree.sol#258)
MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/themoneytree.sol#212-294)
ignores return value by _stakersPool_A_B.remove(_referrer) (contracts/
themoneytree.sol#259)
MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/themoneytree.sol#212-294)
ignores return value by _winnerList.add(_referrer) (contracts/themoneytree.sol#269)
MoneyTree.distrubuteStep02(uint256) (contracts/themoneytree.sol#368-468) ignores return
value by _stakersTotal.remove(v.recipient) (contracts/themoneytree.sol#420)
MoneyTree.distrubuteStep02(uint256) (contracts/themoneytree.sol#368-468) ignores return
value by _stakersPoolA.remove(v.recipient) (contracts/themoneytree.sol#421)
MoneyTree.distrubuteStep02(uint256) (contracts/themoneytree.sol#368-468) ignores return
value by _stakersPool_A_B.remove(v.recipient) (contracts/themoneytree.sol#422)
MoneyTree.distrubuteStep02(uint256) (contracts/themoneytree.sol#368-468) ignores return
value by _winnerList.add(v.recipient) (contracts/themoneytree.sol#430)
MoneyTree.distrubuteStep03(uint256) (contracts/themoneytree.sol#471-572) ignores return
value by _stakersTotal.remove(v.recipient) (contracts/themoneytree.sol#523)
MoneyTree.distrubuteStep03(uint256) (contracts/themoneytree.sol#471-572) ignores return
value by _stakersPoolB.remove(v.recipient) (contracts/themoneytree.sol#524)
MoneyTree.distrubuteStep03(uint256) (contracts/themoneytree.sol#471-572) ignores return
value by _stakersPool_A_B.remove(v.recipient) (contracts/themoneytree.sol#525)
MoneyTree.distrubuteStep03(uint256) (contracts/themoneytree.sol#471-572) ignores return
value by _winnerList.add(v.recipient) (contracts/themoneytree.sol#533)
MoneyTree.distrubuteStep04(uint256) (contracts/themoneytree.sol#575-675) ignores return
value by _stakersTotal.remove(v.recipient) (contracts/themoneytree.sol#627)
MoneyTree.distrubuteStep04(uint256) (contracts/themoneytree.sol#575-675) ignores return
value by _stakersPoolC.remove(v.recipient) (contracts/themoneytree.sol#628)
MoneyTree.distrubuteStep04(uint256) (contracts/themoneytree.sol#575-675) ignores return
value by _winnerList.add(v.recipient) (contracts/themoneytree.sol#636)
MoneyTree.distrubuteStep05(uint256) (contracts/themoneytree.sol#678-743) ignores return
value by _stakersTotal.remove(v.winnerAddress) (contracts/themoneytree.sol#705)
MoneyTree.distrubuteStep05(uint256) (contracts/themoneytree.sol#678-743) ignores return
```

```
value by _stakersPoolA.remove(v.winnerAddress) (contracts/themoneytree.sol#706)
MoneyTree.distrubuteStep05(uint256) (contracts/themoneytree.sol#678-743) ignores return
value by _stakersPoolB.remove(v.winnerAddress) (contracts/themoneytree.sol#707)
MoneyTree.distrubuteStep05(uint256) (contracts/themoneytree.sol#678-743) ignores return
value by _stakersPool_A_B.remove(v.winnerAddress) (contracts/themoneytree.sol#708)
MoneyTree.distrubuteStep05(uint256) (contracts/themoneytree.sol#678-743) ignores return
value by _winnerList.add(v.winnerAddress) (contracts/themoneytree.sol#718)
MoneyTree.payWinner(address) (contracts/themoneytree.sol#944-985) ignores return value
by _stakersTotal.remove(_user) (contracts/themoneytree.sol#952)
MoneyTree.payWinner(address) (contracts/themoneytree.sol#944-985) ignores return value
by _stakersPoolA.remove(_user) (contracts/themoneytree.sol#953)
MoneyTree.payWinner(address) (contracts/themoneytree.sol#944-985) ignores return value
by _stakersPoolB.remove(_user) (contracts/themoneytree.sol#954)
MoneyTree.payWinner(address) (contracts/themoneytree.sol#944-985) ignores return value
by _stakersPoolC.remove(_user) (contracts/themoneytree.sol#955)
MoneyTree.payWinner(address) (contracts/themoneytree.sol#944-985) ignores return value
by _stakersPool_A_B.remove(_user) (contracts/themoneytree.sol#956)
MoneyTree.payWinner(address) (contracts/themoneytree.sol#944-985) ignores return value
by _winnerList.add(_user) (contracts/themoneytree.sol#964)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
MoneyTree.setPoolStartTime(uint256) (contracts/themoneytree.sol#186-190) should emit an
event for:
- poolStartTime = _poolStartTime (contracts/themoneytree.sol#188)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-
arithmetic
INFO:Detectors:
Reentrancy in MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/
themoneytree.sol#212-294):
External calls:
- IERC20(token).safeTransferFrom(_sender,address(this),_amount) (contracts/
themoneytree.sol#225)
State variables written after the call(s):
- epochDepositAmount[currentEpoch] += _amount (contracts/themoneytree.sol#226)
- addUserToGroupCurrentEpochList(_sender) (contracts/themoneytree.sol#237)
- epochUserIndex[_currentEpoch][_user] = epochUsersByGroup[_currentEpoch]
[_userGroup].length (contracts/themoneytree.sol#992)
- removeUserFromGroupCurrentEpochList(_referrer) (contracts/themoneytree.sol#261)
- epochUserIndex[_currentEpoch][lastUserAddress] = userIndex (contracts/
themoneytree.sol#1007)
- delete epochUserIndex[_currentEpoch][_user] (contracts/themoneytree.sol#1009)
- addUserToGroupCurrentEpochList(_sender) (contracts/themoneytree.sol#237)
- epochUsersByGroup[_currentEpoch][_userGroup].push(_user) (contracts/
themoneytree.sol#994)
- removeUserFromGroupCurrentEpochList(_referrer) (contracts/themoneytree.sol#261)
- epochUsersByGroup[_currentEpoch][_userGroup][userIndex] = lastUserAddress
(contracts/themoneytree.sol#1006)
- epochUsersByGroup[_currentEpoch][_userGroup].pop() (contracts/
themoneytree.sol#1012)
- addUserToGroupCurrentEpochList(_sender) (contracts/themoneytree.sol#237)
- isUserInEpochList[_currentEpoch][_user] = true (contracts/themoneytree.sol#993)
- removeUserFromGroupCurrentEpochList(_referrer) (contracts/themoneytree.sol#261)
- isUserInEpochList[_currentEpoch][_user] = false (contracts/themoneytree.sol#1010)
- winnerGroup[_referrer] = _group (contracts/themoneytree.sol#270)
Reentrancy in MoneyTree.distrubuteStep05(uint256) (contracts/themoneytree.sol#678-743):
External calls:
- IERC20(token).safeTransfer(v.winnerAddress,v.winnerPayment) (contracts/
themoneytree.sol#710)
- IERC20(token).safeTransfer(v.winnerAddress,v.distributeAmountLottery) (contracts/
themoneytree.sol#729)
State variables written after the call(s):
- isEpochDistributed[currentEpoch] = true (contracts/themoneytree.sol#740)
- winnerGroup[v.winnerAddress] = _winnerGroup (contracts/themoneytree.sol#719)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-2
INFO:Detectors:
Reentrancy in MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/
themoneytree.sol#212-294):
External calls:
- IERC20(token).safeTransferFrom(_sender,address(this),_amount) (contracts/
themoneytree.sol#225)
- IERC20(token).safeTransfer(_referrer,referrerMaxPayout - receivedAmount) (contracts/
themoneytree.sol#273)
Event emitted after the call(s):
- ReferrerPaymentPaid(_referrer,referrerMaxPayout - receivedAmount) (contracts/
themoneytree.sol#277)
Reentrancy in MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/
themoneytree.sol#212-294):
External calls:
- IERC20(token).safeTransferFrom(_sender,address(this),_amount) (contracts/
themoneytree.sol#225)
- IERC20(token).safeTransfer(_referrer,depositSize * 2) (contracts/
themoneytree.sol#282)
```

```
Event emitted after the call(s):
- ReferrerPaymentPaid(_referrer,depositSize * 2) (contracts/themoneytree.sol#286)
Reentrancy in MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/
themoneytree.sol#212-294):
External calls:
- IERC20(token).safeTransferFrom(_sender,address(this),_amount) (contracts/
themoneytree.sol#225)
- IERC20(token).safeTransfer(_referrer,referrerMaxPayout - receivedAmount) (contracts/
themoneytree.sol#273)
- IERC20(token).safeTransfer(_referrer,depositSize * 2) (contracts/
themoneytree.sol#282)
Event emitted after the call(s):
- Deposited(_sender,_group,_amount,_referrer) (contracts/themoneytree.sol#291)
Reentrancy in MoneyTree.distrubuteStep01(uint256) (contracts/themoneytree.sol#321-365):
External calls:
- IERC20(token).safeTransfer(v.recipient,v.devPaymentAmount) (contracts/
themoneytree.sol#352)
Event emitted after the call(s):
- DevBonusPaid(v.recipient,v.devPaymentAmount) (contracts/themoneytree.sol#354)
Reentrancy in MoneyTree.distrubuteStep01(uint256) (contracts/themoneytree.sol#321-365):
External calls:
- IERC20(token).safeTransfer(v.recipient,v.distributeAmountDev - v.devPaymentAmount *
10) (contracts/themoneytree.sol#348)
- IERC20(token).safeTransfer(v.recipient,v.devPaymentAmount) (contracts/
themoneytree.sol#352)
- IERC20(token).safeTransfer(tradingAccount,v.distributeAmountTrading) (contracts/
themoneytree.sol#358)
Event emitted after the call(s):
- TradingAccountFunded(tradingAccount,v.distributeAmountTrading) (contracts/
themoneytree.sol#362)
Reentrancy in MoneyTree.distrubuteStep05(uint256) (contracts/themoneytree.sol#678-743):
External calls:
- IERC20(token).safeTransfer(v.winnerAddress,v.winnerPayment) (contracts/
themoneytree.sol#710)
- IERC20(token).safeTransfer(v.winnerAddress,v.distributeAmountLottery) (contracts/
themoneytree.sol#729)
Event emitted after the call(s):
- LotteryBonusPaid(v.winnerAddress,v.winnerPayment) (contracts/themoneytree.sol#723)
- LotteryBonusPaid(v.winnerAddress,v.distributeAmountLottery) (contracts/
themoneytree.sol#731)
Reentrancy in MoneyTree.payWinner(address) (contracts/themoneytree.sol#944-985):
```

```
External calls:
- IERC20(token).safeTransfer(_user,userMaxPayout - receivedAmount) (contracts/
themoneytree.sol#967)
Event emitted after the call(s):
- PoolBonusPaid(_user,userMaxPayout - receivedAmount) (contracts/themoneytree.sol#971)
Reentrancy in MoneyTree.payWinner(address) (contracts/themoneytree.sol#944-985):
External calls:
- IERC20(token).safeTransfer(_user,depositSize) (contracts/themoneytree.sol#977)
Event emitted after the call(s):
- PoolBonusPaid(_user,depositSize) (contracts/themoneytree.sol#983)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-3
INFO:Detectors:
MoneyTree.setPoolStartTime(uint256) (contracts/themoneytree.sol#186-190) uses timestamp
for comparisons
Dangerous comparisons:
- _poolStartTime < block.timestamp (contracts/themoneytree.sol#187)
MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/themoneytree.sol#212-294)
uses timestamp for comparisons
Dangerous comparisons:
- userInfo[_referrer].lastEpochAddReferrals == currentEpoch (contracts/
themoneytree.sol#240)
MoneyTree.isTimeInWindow(uint256) (contracts/themoneytree.sol#911-915) uses timestamp
for comparisons
Dangerous comparisons:
- _time < poolStartTime (contracts/themoneytree.sol#912)
- diff - (diff / 604800) * 604800 < 86400 (contracts/themoneytree.sol#914)
MoneyTree.getEpoch(uint256) (contracts/themoneytree.sol#918-922) uses timestamp for
comparisons
Dangerous comparisons:
- _time < poolStartTime (contracts/themoneytree.sol#919)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-
timestamp
INFO:Detectors:
MoneyTree.distrubuteStep01(uint256) (contracts/themoneytree.sol#321-365) compares to a
boolean constant:
-isEpochDistributed[currentEpoch] == true (contracts/themoneytree.sol#329)
MoneyTree.distrubuteStep02(uint256) (contracts/themoneytree.sol#368-468) compares to a
boolean constant:
-userInfo[v.recipient].winner == true (contracts/themoneytree.sol#413)
MoneyTree.distrubuteStep03(uint256) (contracts/themoneytree.sol#471-572) compares to a
```

```
boolean constant:
	-userInfo[v.recipient].winner == true (contracts/themoneytree.sol#516)
MoneyTree.distrubuteStep04(uint256) (contracts/themoneytree.sol#575-675) compares to a
boolean constant:
	-userInfo[v.recipient].winner == true (contracts/themoneytree.sol#620)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-
equality
INFO:Detectors:
MoneyTree.processRandomness(uint256,uint256,uint256) (contracts/
themoneytree.sol#1028-1033) has costly operations inside a loop:
	- nonce ++ (contracts/themoneytree.sol#1030)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-
operations-inside-a-loop
INFO:Detectors:
MoneyTree.deposit(MoneyTree.Group,uint256,address) (contracts/themoneytree.sol#212-294)
has a high cyclomatic complexity (18).
MoneyTree.getStakersList(MoneyTree.Group,uint256,uint256) (contracts/
themoneytree.sol#857-891) has a high cyclomatic complexity (13).
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#cyclomatic-
complexity
INFO:Detectors:
Pragma version0.8.19 (contracts/themoneytree.sol#3) necessitates a version too recent
to be trusted. Consider deploying with 0.8.18.
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-
versions-of-solidity
INFO:Detectors:
Parameter MoneyTree.initialize(address[],address)._devs (contracts/
themoneytree.sol#171) is not in mixedCase
Parameter MoneyTree.initialize(address[],address)._tradingAccount (contracts/
themoneytree.sol#171) is not in mixedCase
Parameter MoneyTree.setPoolStartTime(uint256)._poolStartTime (contracts/
themoneytree.sol#186) is not in mixedCase
Parameter MoneyTree.setGroupsInfo(MoneyTree.Group[],MoneyTree.GroupInfo[])._groups
(contracts/themoneytree.sol#192) is not in mixedCase
Parameter MoneyTree.setGroupsInfo(MoneyTree.Group[],MoneyTree.GroupInfo[])._infos
(contracts/themoneytree.sol#192) is not in mixedCase
Parameter MoneyTree.setLinkToken(address)._linkToken (contracts/themoneytree.sol#205)
is not in mixedCase
Parameter MoneyTree.deposit(MoneyTree.Group,uint256,address)._group (contracts/
themoneytree.sol#212) is not in mixedCase
```

```
Parameter MoneyTree.deposit(MoneyTree.Group,uint256,address)._amount (contracts/
themoneytree.sol#212) is not in mixedCase
Parameter MoneyTree.deposit(MoneyTree.Group,uint256,address)._referrer (contracts/
themoneytree.sol#212) is not in mixedCase
Parameter MoneyTree.distrubuteStep01(uint256)._requestId (contracts/
themoneytree.sol#321) is not in mixedCase
Parameter MoneyTree.distrubuteStep02(uint256)._requestId (contracts/
themoneytree.sol#368) is not in mixedCase
Parameter MoneyTree.distrubuteStep03(uint256)._requestId (contracts/
themoneytree.sol#471) is not in mixedCase
Parameter MoneyTree.distrubuteStep04(uint256)._requestId (contracts/
themoneytree.sol#575) is not in mixedCase
Parameter MoneyTree.distrubuteStep05(uint256)._requestId (contracts/
themoneytree.sol#678) is not in mixedCase
Parameter MoneyTree.winners(uint256)._index (contracts/themoneytree.sol#746) is not in
mixedCase
Parameter MoneyTree.winnerListContains(address)._user (contracts/themoneytree.sol#750)
is not in mixedCase
Parameter MoneyTree.getUserInfo(address)._user (contracts/themoneytree.sol#767) is not
in mixedCase
Parameter MoneyTree.getWinnerGroup(address)._user (contracts/themoneytree.sol#793) is
not in mixedCase
Parameter MoneyTree.getEpochDepositAmount(uint256)._epoch (contracts/
themoneytree.sol#798) is not in mixedCase
Parameter MoneyTree.stakersByGroup(MoneyTree.Group,uint256)._group (contracts/
themoneytree.sol#809) is not in mixedCase
Parameter MoneyTree.stakersByGroup(MoneyTree.Group,uint256)._index (contracts/
themoneytree.sol#809) is not in mixedCase
Parameter MoneyTree.stakersContainsByGroup(MoneyTree.Group,address)._group (contracts/
themoneytree.sol#825) is not in mixedCase
Parameter MoneyTree.stakersContainsByGroup(MoneyTree.Group,address)._user (contracts/
themoneytree.sol#825) is not in mixedCase
Parameter MoneyTree.stakersLengthByGroup(MoneyTree.Group)._group (contracts/
themoneytree.sol#841) is not in mixedCase
Parameter MoneyTree.getStakersList(MoneyTree.Group,uint256,uint256)._group (contracts/
themoneytree.sol#857) is not in mixedCase
Parameter MoneyTree.getEpochUsersByGroup(MoneyTree.Group)._group (contracts/
themoneytree.sol#893) is not in mixedCase
Parameter MoneyTree.getEpochUserIndex(address)._user (contracts/themoneytree.sol#899)
is not in mixedCase
Function MoneyTree._isUserInEpochList(address) (contracts/themoneytree.sol#905-908) is
```

```
not in mixedCase
Parameter MoneyTree._isUserInEpochList(address)._user (contracts/themoneytree.sol#905)
is not in mixedCase
Parameter MoneyTree.isTimeInWindow(uint256)._time (contracts/themoneytree.sol#911) is
not in mixedCase
Parameter MoneyTree.getEpoch(uint256)._time (contracts/themoneytree.sol#918) is not in
mixedCase
Parameter MoneyTree.getRequestStatus(uint256)._requestId (contracts/
themoneytree.sol#925) is not in mixedCase
Parameter MoneyTree.fulfillRandomWords(uint256,uint256[])._requestId (contracts/
themoneytree.sol#932) is not in mixedCase
Parameter MoneyTree.fulfillRandomWords(uint256,uint256[])._randomWords (contracts/
themoneytree.sol#932) is not in mixedCase
Parameter MoneyTree.payWinner(address)._user (contracts/themoneytree.sol#944) is not in
mixedCase
Parameter MoneyTree.addUserToGroupCurrentEpochList(address)._user (contracts/
themoneytree.sol#988) is not in mixedCase
Parameter MoneyTree.removeUserFromGroupCurrentEpochList(address)._user (contracts/
themoneytree.sol#998) is not in mixedCase
Parameter MoneyTree.processRandomness(uint256,uint256,uint256)._requestId (contracts/
themoneytree.sol#1028) is not in mixedCase
Parameter MoneyTree.processRandomness(uint256,uint256,uint256)._k (contracts/
themoneytree.sol#1028) is not in mixedCase
Parameter MoneyTree.processRandomness(uint256,uint256,uint256)._size (contracts/
themoneytree.sol#1028) is not in mixedCase
Parameter MoneyTree.calculateGroupDistribution(uint256)._totalAmount (contracts/
themoneytree.sol#1036) is not in mixedCase
Variable MoneyTree._stakersPool_A_B (contracts/themoneytree.sol#102) is not in
mixedCase
Variable MoneyTree.s_requests (contracts/themoneytree.sol#119) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-
solidity-naming-conventions
INFO:Detectors:
Variable MoneyTree._stakersPoolA (contracts/themoneytree.sol#97) is too similar to
MoneyTree._stakersPoolB (contracts/themoneytree.sol#98)
Variable MoneyTree._stakersPoolA (contracts/themoneytree.sol#97) is too similar to
MoneyTree._stakersPoolC (contracts/themoneytree.sol#99)
Variable MoneyTree._stakersPoolB (contracts/themoneytree.sol#98) is too similar to
MoneyTree._stakersPoolC (contracts/themoneytree.sol#99)
Variable MoneyTree.distributeAmountPoolAStorage (contracts/themoneytree.sol#76) is too
similar to MoneyTree.distributeAmountPoolBStorage (contracts/themoneytree.sol#77)
```

```
Variable MoneyTree.distributeAmountPoolAStorage (contracts/themoneytree.sol#76) is too
similar to MoneyTree.distributeAmountPoolCStorage (contracts/themoneytree.sol#78)
Variable MoneyTree.distributeAmountPoolBStorage (contracts/themoneytree.sol#77) is too
similar to MoneyTree.distributeAmountPoolCStorage (contracts/themoneytree.sol#78)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-
too-similar
INFO:Detectors:
MoneyTree.slitherConstructorVariables() (contracts/themoneytree.sol#14-1046) uses
literals with too many digits:
- callbackGasLimit = 500000 (contracts/themoneytree.sol#91)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-
digits
INFO:Detectors:
MoneyTree.callbackGasLimit (contracts/themoneytree.sol#91) should be constant
MoneyTree.numWords (contracts/themoneytree.sol#94) should be constant
MoneyTree.requestConfirmations (contracts/themoneytree.sol#92) should be constant
MoneyTree.vrfV2Wrapper (contracts/themoneytree.sol#84) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-
variables-that-could-be-declared-constant
INFO:Detectors:
MoneyTree.keeper (contracts/themoneytree.sol#86) should be immutable
MoneyTree.token (contracts/themoneytree.sol#71) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-
variables-that-could-be-declared-immutable
INFO:Slither:. analyzed (13 contracts with 88 detectors), 131 result(s) found
```

Guard