Report on TronAdz

Created By: Grim Reaper

Telegram: @grimreaper916

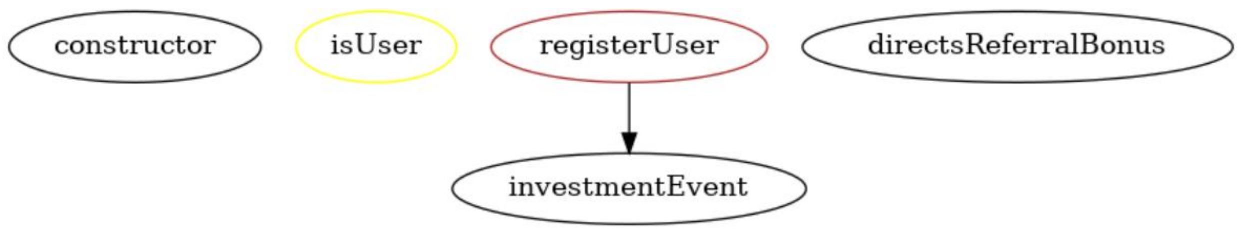
Disclaimer

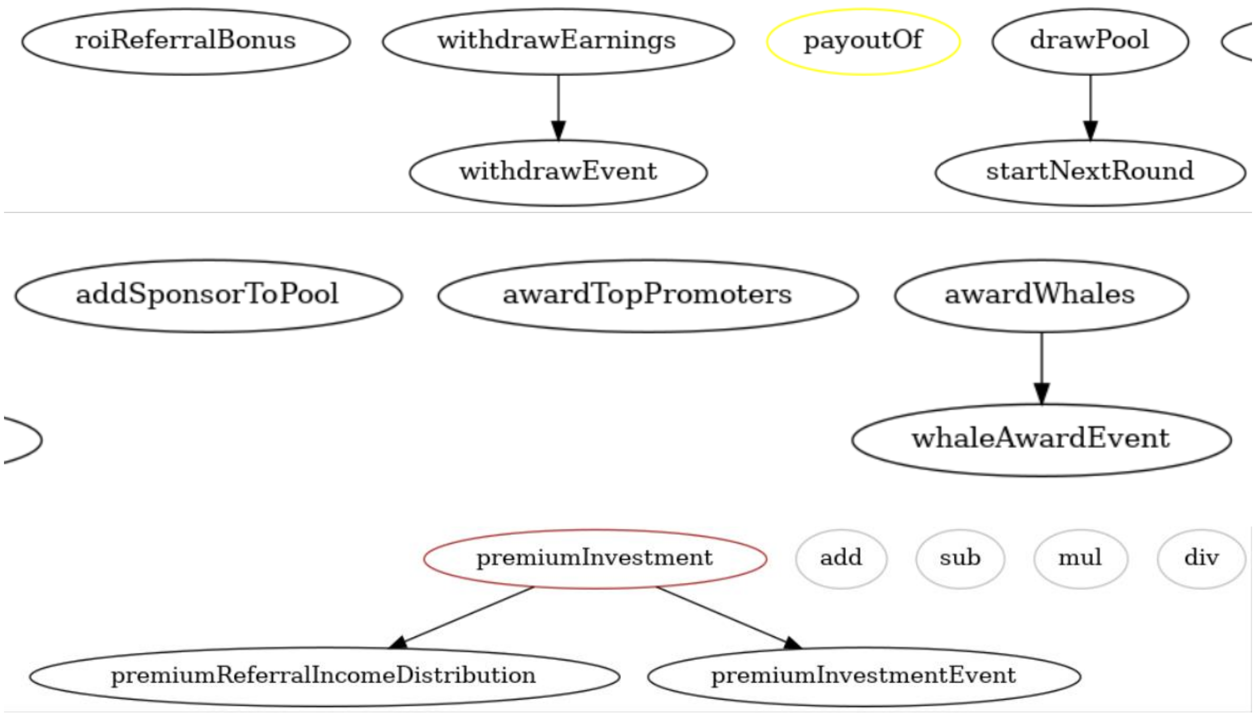
This text is not a call to participate in the project and it's only a description of the smart contract work at the specified address. Remember that you do all the financial actions only at your own risk.

Description

Tronadz is a contract which creates a HYIP platform based on ponzi scheme. It offers its users to deposit TRON and gain profit on a daily basis. There are supposedly four rounds of investment cycles with increasing amount of pool limit for each round. The contract utilizes the standard SafeMath library for arithmetics and one user defined library called DataStructs for storing user data.

Contract Structure





Legend**:**

- Red: Send to external address
- Blue: Constant function
- Yellow: View
- Green: Pure
- Orange: Call
- Purple: Transfer
- Lilac: Payable

Variables

```
address owner;

address  masterAccount;

uint256  houseFee = 5; (private)

uint256  poolTime = 24 hours; (private)

uint256  dailyWinPool = 5; (private)

uint256  whalepoolPercentage = 25; (private)

uint256  incomeTimes = 30; (private)

uint256  incomeDivide = 10; (private)

uint256  total_withdraw;

uint256  roundID;

uint256  currUserID;

uint256[4]  awardPercentage;

struct Leaderboard {

            uint256 amt;

             address addr; }

Leaderboard[4]  topSponsors;

Leaderboard[4]  lastTopSponsors;

uint256[4]  lastTopSponsorsWinningAmount;

address[]  whales;
```

mapping (uint => uint) CYCLE_LIMIT;

mapping (address => bool) isWhale;

mapping (uint => address) userList;

mapping (uint256 => DataStructs.DailyRound) round;

mapping (address => DataStructs.User) player;

mapping (address =>uint256) playerTotEarnings;

mapping (address => mapping (uint256 => DataStructs.PlayerDailyRounds)) plyrRnds_;

Functions

Line 52-77: constructor() – Sets owner account and initializes each cycles limit with according values

Line 79-82: isUser() – Returns true or false based on user participation

Line 90-93: isMinimumAmount() – Modifier which restricts minimum amount to deposit

Line 98-101: isallowedValue() – Modifier which limits certain values for deposits

Line 106-109: onlyOwner() – Restricts usage only to owner

Line 117-194: registerUser() – Registers a user who deposits into the contract with a certain referral id. If the deposit amount is higher than a certain value, he is added to a whale list. 5% of the deposit is transferred to owner as housefee

Line 197-230: directsReferralBonus() (Private) – Function which calculates the multi tier direct referral bonus

Line 234-280: roiReferralBonus() (Private) - Function to manage the referral commission from the daily ROI

Line 285-379: withdrawEarnings() – Function which processes the users withdrawals. It also calculates sponsor bonus, pool payout, whale bonus and premium adz payout.

Line 381-391: payoutOf() (view) – Calculates daily profit of the user

Line 395-425: startNextRound() (Private) – It starts next cycle after calling whales and top promoters payout function

Line 428-544: addSponsorToPool() (Private) – Adds top 3 sponsors to a list according to the amount raised

Line 546-573: awardTopPromoters() (Private) – Pays top 3 promoters from the amount inside sponsor pool

Line 575-586: awardWhales() (Private) – Pays big depositors from the amount inside whale pool

Line 588-599: premiumInvestment() – Accepts deposits from the user

Line 601-635: premiumReferralIncomeDistribution() (Private) – Calculates multi tier referral bonus from premiumInvestment()

Line 638-640: drawPool() – OnlyOwner function which can start next round

Best Practices

The code has written according to all Secure Development Recommendations. The libraries used inside are all standard and secure. It was deployed successfully on testnet (Shasta) and tested.

Critical Severity

No errors or vulnerabilities affecting the described functionality of the contract have been detected. No backdoors or overflows are present in the contract

Medium Severity

- No fallback function to handle direct payments to the contract
- Private variables mentioned are open for anyone
- *Line 385*: Multiplication done before division. Possible precision errors in calculating the value
- *Line 638*: Owner has excess privilege to start next round manually without following the time schedule

Low Severity

- SafeMath operations are used unnecessarily. It can be avoided in places with no chance of overflows
- *Line 7*: Unnecessary masterAccount variable. Owner variable can be used in its place
- *Line 588-635*: Potentially unwanted functions premiumInvestment() and premiumReferralIncomeDistribution(). They does not interact with the

contracts actual deposit function and amounts deposited via that function are not counted.

Other Analysis

- _eth variable name is presumably copied from Ethereum smart contract. It could be changed to tron for readability purposes
- *Line 186,374,596*: Instead of storing into another variable, msg.sender can be directly used for gas optimization
- *Line 305*: if clauses could be nested to compress code and optimize gas usage
- *Line 581*: Loop control variable is dynamic, which can lead to a costly loop in the future