TRON GO SMART CONTRACT

PEAK AUDIT

AUDITING AND CODE REVIEW

Auditor: *@peakaudit* (Telegram)

Date: 2021-07-23

CONTRACT DETAILS

Project website	**https://trongo.space/**
Contract Address	**TM9dFRzxUhuiH2tGpzpiYmjLQxCcp6Yr39**
Contract Url	**https://tronscan.org/#/contract/TM9dFRzxUhuiH2tGpzpiYmjLQxCcp6Yr39/code**
Category	**High Risk ROI Smart Contract**
Language and version	**Solidity 0.5.10**
Network	**Tron Blockchain**
Audit Type	**Basic Audit – Functional Report**

ABSTRACT

TRON GO is a ROI DAPP Smart Contract, fully decentralized and operating on the Tron Blockchain. In the following report the main features will be analyzed and the source code will be checked for security bugs. Main mechanics will be reported.

Please note:

- This audit won’ t analyze high level mechanics or express considerations about the sustainability of the ROI DAPP.
- This audit is technical, we don’ t give f inancial /investment advises.
- The audit is related ONLY to the Smart Contract deployed at the address indicated. Always check the address of the contract before interacting with it.

FEATURES

Here are reported the features of the smart contract as declared in the source file. According to analysis of TRON GO community it is designed in a way that the ROI will increase on a fix limit when the balance increase in project , TRON GO will have starting 4.5% to 9.5% ROI up to 300% . When TRON GO will reach 150K TRX in contract balance it ROI will increase to 5.5%. TRON GO balance will go on to increase 1% on every 150K TRX in balance up to maximum of 9.5% at 750 K TRX contract balance.

The minimum deposit amount is 100 TRX (plus ~35 TRX for the first deposit and ~15-20 TRX for the following deposits). There is no withdraw limit (the

withdraw consumes ~30 TRX). There is a single *withdraw()* function for both the dividends and the referral rewards.

From the code point of view they are added together in the *withdraw* function

before the transfer.

```
*   [SMART CONTRACT DETAILS]
*
*   - ROC(return of contribution): 4.5% to 9.5 % every 24h – Max 300% for every deposit
*   - Minimal deposit: 100 TRX, no maximal limit.
*   - Single button withdraw for dividends and cumulated referral bonus
*   - Withdraw any time, but with the limit of max1 withdraw every 24h. No max withdraw limit.
```

FEATURES

The affiliate program is composed of 5 levels. The commissions are paid for every deposit. The *_referralPayout()* function is called inside *deposit()* . If the sponsor is not a valid / registered user (an user that joined before) the contract sets the admin Address as referral. The declared percentages has been verified in the source.

The withdraw function is used for both the dividends (ROI computation payouts) and referral bonus. All the eligible rewards are summed together and then sent to the user wallet.

```
*    [AFFILIATE PROGRAM]
*
*    - 5-level referral commission: 7%- 3%- 2%- 1%- 4%- Earn more from the last line!
*    - If you don't have a sponsor, you can still join: the admin (adminAddress) will be your sponsor
*    - Your sponsor must be a registered user (another participant who deposited), otherwise the admin will be your sponsor
*    - Once joined you cannot change your sponsor
*
```

FEATURES

The fees applied by the smart contract are 2 : developer fee, marketing fee . Every fee is associated to a different wallet address. All the fees are sent af ter a deposit and are calculated on the deposited amount. There are no other fees.

The registered addresses on the deployed version of the contract (provided URL) are:

- devAddress = TNTo8tfnKoSXNh4Luc1NYwr3LLMDJaGHeh
- marketingAddress = TYpCvtH4a4HhfTUhbnhsUFCWZc9aaDK2Ee

```
*     [FUNDS DISTRIBUTION]
*
*     - 68%Platform main balance, participants payouts
*     - 7.5%Advertising and promotion expenses
*     - 7.5% Developer fee , Support work, technical functioning
*     - 17%Affiliate program bonuses
```

FEATURES – CONCLUSION

- Dividends and referral bonus are paid from deposits of users. The Smart Contract provides the opportunity to obtain a 300%

  All dividends are calculated at the moment of the withdraw request, allowed each 24h. This calculation updates all the information associate d to each deposit

- Each deposit is kept separately in the contract, in order to maintain the payment amount (history) for each one

- The applied fees (17%) are fair and the referral program balanced, with an incentive to create a team (more commissions on 5th level than the others

- According to analysis of TRON GO community it is designed in a way that the ROI will increase on a fix limit when the balanc e increase in project , TRON GO will have starting 4.5% to 9.5 % ROI up to 300% . When TRON GO will reach 150K TRX in contract balance it R OI will increase to 5.5%. TRON GO balance will go on to increase 1% on every 100K TRX in balance up to maximum of 9.5 % at 750K TRX contract balance.

CODE AUDIT - CONSTANTS

The declared percentages for fees and referrals program are correct. The minimum deposit amount and the max amount withdrawable per day are correct. All the ROI parameters are the ones declared.

```
    // Operating costs
    uint256 constant public MARKETING_FEE = 750;
    uint256 constant public DEV_FEE = 750;
    uint256 constant public PERCENTS_DIVIDER = 1000;
    // Referral percentages
    uint8 public constant FIRST_REF = 7;
    uint8 public constant SECOND_REF = 3;
    uint8 public constant THIRD_REF = 2;
    uint8 public constant FOURTH_REF = 1;
    uint8 public constant FIFTH_REF = 4;
    // Limits
    uint256 public constant DEPOSIT_MIN_AMOUNT = 100 trx;
    // Before reinvest
    uint256 public constant WITHDRAWAL_DEADTIME = 1 days;
    // Max ROC days and related MAX ROC (Return of contribution)
    uint8 public constant CONTRIBUTION_DAYS = 100;
    uint256 public constant CONTRIBUTION_PERC = 300;
```

CODE AUDIT - CONSTRUCTOR

The constructor initializes the addresses used for the fees and the referral levels. No issues found in this part

```
    constructor(address payable marketingAddr, address payable adminAddr, address payable devAddr) public {
        require(!isContract(marketingAddr) &&!isContract(adminAddr) &&!isContract(devAddr));

        marketingAddress = marketingAddr;
        adminAddress = adminAddr;
        devAddress = devAddr;
        owner = msg.sender;

        // Add referral bonuses (max 8 levels) - We use 5 levels
        referral_bonuses.push(10 * FIRST_REF);
        referral_bonuses.push(10 * SECOND_REF);
        referral_bonuses.push(10 * THIRD_REF);
        referral_bonuses.push(10 * FOURTH_REF);
        referral_bonuses.push(10 * FIFTH_REF);
    }
```

CODE AUDIT – FUNCTIONS

```
function deposit(address _referral) external payable {
    require(!isContract(msg.sender) &&msg.sender == tx.origin);
    require(!isContract(_referral));
    require(msg.value >= 1e8, "Zero amount");
    require(msg.value >= DEPOSIT_MIN_AMOUNT, "Deposit is below minimum amount");
    Player storage player = players[msg.sender];
    require(player.deposits.length < 1500, "Max 1500 deposits per address");
    // Check and set referral
    _setReferral(msg.sender, _referral);
    // Create deposit
    player.deposits.push(PlayerDeposit({
        amount: msg.value,
        totalWithdraw: 0,
        time: uint256(block.timestamp)
    }));
    // Add new user if this is first deposit
    if(player.total_contributed == 0x0){
        total_investors += 1;
    }
    player.total_contributed += msg.value;
    total_contributed += msg.value;
    // Generate referral rewards
    _referralPayout(msg.sender, msg.value);
    // Pay fees
    _feesPayout(msg.value);
    emit Deposit(msg.sender, msg.value);
}
```

No issues found in the *deposit()* function. The function can be called only by addresses and not other smar t contracts and all the checks are done at the top of the function. There is a fur ther requirement of maximum 1500 deposits per address.

The function f i rst sets the referral address (upline) calling the private function *_ set Referral(),* then creates the deposit entr y and then pays the upline and the contract fees with *_ referral Payout()* and *_ fees Payout()* private functions respectively.

ISSUES SUMMARY

In the following table is reported the summary of the issues / problems found in the audited Smart Contract

Type	Found	Description
Critical Issues *Critical – High Severity*	**0**	Critical and harmful access for owners, user block ability, bugs and vulnerabilities that enable theft of funds, lock access to funds without possibility to restore it, or lead to any other loss of funds to be transferred to any party
Errors, Bugs and Warnings *Medium – Low Severity*	**0**	Bugs that can negatively affect the usability of a program, errors that can trigger a contract failure, lack of necessary security precautions, other warnings for owner and users, warning codes that are valid but the compiler thinks are suspicious
Optimizations *Low Severity*	**1**	Methods to decrease the cost of transactions in the Smart Contracts, cleaner code, memory or logic optimizations
Recommendations *Very Low Severity*	**0**	Hint and tips to improve contract functionality and trustworthy

ISSUES NOTES

1. <u>Optimization Issue (low severity):</u> loop on a dynamic variable when withdrawing:

When withdrawing the cycle will operate on each deposit record. With the increasing of user deposits, the execution time of the withdraw function increases and so the associated fees. In case of exceeding the allowed Tron limit of 100 TRX configured in the Tron Web calls (frontend), withdraw can become impossible. This could affect users with hundreds of deposits. In any case the Smart Contract limits the maximum number of deposits / redeposits to 1500, more than enough for a fair use.

<u>Suggestions for the users:</u>

Users are advised to make less but bigger deposits than more but smaller deposits, to reduce fees when withdrawing.

Freezing TRX to obtain Energy and Bandwidth is a good idea to reduce or even compensate completely the transaction fees of the Smart Contract

<u>Suggestions for the developers:</u>

Increase the Tron Limit in the frontend if needed. Keep the maximum limit to 1000 TRX in the deployed contract

Pay some of the fees from the contract instead of 100 % from the user (e.g. 60 % contract 40 % user). This will be good for medium-sized deposits

CONCLUSIONS

- This audit is only to the Smart Contract at the given address
- **No vulnerabilities, no backdoors and no scams found in the TRONGO Smart Contract**
- The code was compiled and tested using Shasta Testnet, compiler version 0.10 without errors (optimizer 200 runs). The same code is deployed on main - net and currently used with the provided website. Always make sure you are interacting with the contract written in the " Contract Detai ls" slide!

The audit makes no statements or warranties about the suitability or sustainability of the business modelor regulator y regime for the business model. Do take in consideration that you are doing a l l f inancial actions and t ransactions at your own r i sk especially when dealing with high risk projects like ROI dApps /games.

Auditor: @peakaudit