Report on QZX

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

The QZX project developed QZX token which is distributed among project participants in 3 stages. There is an option to invest on the contract and acquire QZX tokens, which can be sold back to contract itself for tron. For each stage, QZX price will be different.

Variables

Passive

mapping (address => Deposit []) public deposits; uint internal depositTerm;

uint internal oneWeek;

_ReferralStorage internal refStorage;

Token internal token;

_Tokemonics internal tokemonics;

_ LevelsAndTurnovers internal levelsAndTurnovers;

```
struct Deposit {
        uint amount;
        uint time;
        uint8 countOfweeks;
}
```

ReferralStorage

mapping(address=>address) public referrals;

mapping(address => address []) public referrers;

Passive public passive;

LevelsAndTurnovers

_Passive public passive;

_ReferralStorage public refStorage;

TRC20

string public constant name = "QZX";

string public constant symbol = "QZX";

uint8 public constant decimals = 8;

mapping (address => uint256) private _balances;

mapping (address => mapping (address => uint256)) private _allowances;

mapping (address => uint) refTokens;

mapping (address => uint) investTokens;

uint public reservedTokens;

uint256 private _totalSupply;

_Passive public passive;

Tokenomics

```
struct Stage {
        uint lessAmount;
        uint largeAmount;
}
Token public token;
uint public tokensForRef;
Stage [] public stages; uint public currentStage;
address passive;
```

Functions

(Standard libraries, Interfaces and contracts are omitted)

Passive

Line 72: getCountofDeposits() – Returns number of deposits from a user

Line 76: Fallback function

Line 90: payout() – Payout function for deposited users

Line 126: register() – Registers referral address with an account

Line 143: investment() – Allows users to deposit into the contract and receive corresponding token amount as reward

Line 388: calculateTokens() – Calculates the number of tokens to be sent to deposited customers

Line 401: calculatePassive() – Calculates passive income of the customer Line 426: getDeposit() – Returns deposits of customers

Line 434: getTimeDeposit() – Returns time of deposit of customer

Line 442: checkActivity() – Returns activity status of customer over a week

ReferralStorage

Line 31: addref() – Binds user to a particular referral address

Line 38: getLine() Returns the referrals in a particular line

Line 44: _line() Calculates the number of referrals in a specific level

Line 66: getReferral() – Returns referrals of a user Line 70: getReferrers() – Returns referrers of a user

LevelsAndTurnovers

Line 19: totalTurnover() – Returns total turnover amount of a user

Line 32: _maxLeg() – Returns amount, its percentage and maxLeg of referrer in 11th line of a user

Line 59: personalTurnover() – Returns personal turnover of a user Line 63: getLevel() – Returns current level where a user is at

TRC20

Line 49: totalSupply() – Total supply of token

Line 53: thisBalance() – Balance of contract

Line 57: tokenPrice() – Current token price of sale Line 62: tokenSell() – Sell token back to the contract

Line 83,87,96,104,109,126,138,153,169,177,189: Standard TRC20 functions

Tokenomics

Line 43: calculateTokens() – Calculate tokens to be send for received tron value in each stage

Line 158: subTokensForRef() – Substracts token amount for referrals

Line 167: subTokenForInvest() – Substracts token amount according to the stages

Best Practices

The code has written according to all Secure Development Recommendations. The libraries used inside are all standard and secure. Code is properly organized and indented

Critical Severity

No critical severity errors were found during the analysis. Compiles and deploys successfully.

Medium Severity

- Line 209, Contract - Passive.sol : Possible logical error (50 instead of 30)

Low Severity

No lower severity errors were detected during the test phase

Other Findings

- Safemath is used excessively. Only use safemath when there is a chance of arithmetic overflow occurs