Report on QZX

Prepared By: Grim Reaper

Telegram: @grimreaper619

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 and contracts are omitted)

Passive

Line 168: getCountofDeposits() – Returns number of deposits from a user

Line 183: getDecimals() – Returns token decimals

Line 187: setToken() – Sets new token address

Line 192: setTokenomics() – Sets new tokenomics address

Line 197: setRefStorage() – Sets new referralstorage address

Line 207: Fallback function

Line 223: payout() – Payout function for deposited users

Line 251: register() – Registers referral address with an account

Line 259: bytesToAddress() – Converts byte value into address

Line 266: investment() – Allows users to deposit into the contract and receive corresponding token amount as reward

Line 489: calculateTokens() – Calculates the number of tokens to be sent to deposited customers

Line 502: calculatePassive() – Calculates passive income of the customer

Line 522: getDeposit() – Returns deposits of customers

Line 530: getTimeDeposit() – Returns time of deposit of customers

Line 534: checkActivity() – Returns activity status of customer over a week

ReferralStorage

Line 577: addref() – Binds user to a particular referral address

Line 584: _1Line() – Returns first level referrers

Line 595: _2Line() – Returns second level referrers

Line 606: _3Line() – Returns third level referrers

Line 617: _4Line() – Returns fourth level referrers

Line 628: _5Line() – Returns fifth level referrers

Line 639: _6Line() – Returns sixth level referrers

Line 650: _7Line() – Returns seventh level referrers

Line 661: _8Line() – Returns eighth level referrers

Line 672: _9Line() – Returns nineth level referrers

Line 683: _10Line() – Returns tenth level referrers

Line 694: _11Line() – Returns eleventh level referrers

Line 705: _12Line() – Returns twelveth level referrers

Line 716: getReferral() – Returns referrals of a user

Line 720: getReferrers() – Returns referrers of a user

LevelsAndTurnovers

Line 738: setpassive() – Sets new passive contract address

Line 742: setRefStorage() – Sets new refstorage contract address

Line 746: totalTurnover() – Returns total turnover amount of a user

Line 760: _maxLeg() – Returns amount, its percentage and maxLeg of referrer in 11th line of a user

Line 782: personalTurnover() – Returns personal turnover of a user

Line 786: getLevel() – Returns current level where a user is at

TRC20

Line 930: setPassive() – Sets new passive contract address

Line 939: totalSupply() – Total supply of token

Line 943: thisBalance() – Balance of contract

Line 947: tokenPrice() – Current token price of sale

Line 953: tokenSell() – Sell token back to the contract

Line 965,969,978,982,987,993,998,1003,1012,1020,1028: Standard TRC20 functions

Line 1036: _burnFrom() – Burn token from a specified address

Tokenomics

Line 1076: subTokensForRef() – Substracts token amount for referrals

Line 1085: subTokenForInvest() – Substracts token amount according to the stages

Line 1170: calculateTokens() – Calculate tokens to be send for received tron value in each stage

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

- Line 251 – There is no checking if the referrer is same as referral. This allows users to make self referred deposits
- Line 187,192,197,202 – Owner has high privilages to change tokenomics and passive contracts anytime

Medium Severity

- Line 207 – Fallback function is commented out
- Line 930 – setPassive function is accessible to public. Better restrict it to onlyOwner
- Line 259,1036 – Functions defined, but not used even once in the contract

Low Severity

- Line 584,595,606,617,628,639,650,661,672,683,694,705 – Use a common function which can call these functions by a given parameter. It will reduce the code length greatly
- Line 80,85,168,187,192,202,266,502,577 – Usage of external instead of public is more promoted
- Line 906,907,1054 – Visibility is not defined

Other Findings

- Line 226 – Storing loop control variable in another variable will reduce gas fee
- Line 266 – Using pre computed values will help reducing gas fee (ex:10*6 can be directly written as 60)
- Line 266 – Use loop to fold the if conditions and reduce size of code
- Safemath is used excessively. Only use safemath when there is a chance of arithmetic overflow occurs