November 2020

Security Audit

Tronner

TYEAp4RuSohdg3FzokBUwE9xbjiWDRTsEz

www.grox.solutions

CRITICAL ISSUES (critical, high severity): **0**

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; high priority unacceptable bugs for deployment at mainnet; critical warnings for owners, customers or investors.

ERRORS, BUGS AND WARNINGS (medium, low severity): **0**

Bugs that can trigger a contract failure, with further recovery only possible through manual modification of the contract state or contract replacement altogether; Lack of necessary security precautions; other warnings for owners and users.

OPTIMIZATION POSSIBILITIES (very low severity): **1**

Possibilities to decrease cost of transactions and data storage of Smart-Contracts.

NOTES AND RECOMMENDATIONS (very low severity): **2**

Tips and tricks, all other issues and recommendations, as well as errors that do not affect the functionality of the Smart-Contract.

Conclusion:

In the Tronner Smart-Contract were found no vulnerabilities and no backdoors. The code was manually reviewed for all commonly known and more specific vulnerabilities.

So Tronner Smart-Contract is safe for use in the main network.

AUDIT RESULT:

Optimization possibilities

1. Recording statistical parameters in the blockchain (very low severity):

List of statistical parameters that also increase the cost of transactions and increase the amount of data stored in the blockchain:

```
    uint256 public totalUsers;
    uint256 public totalInvested;
    uint256 public totalWithdrawn;
    uint256 public totalDeposits;
    uint256 level1;
    uint256 level2;
    uint256 level3;
    uint256 withdrawRef;
```

Recommendation: use events and log this information instead of writing it to the blockchain.

Note: this comment doesn't affect the main functionality of the smart-contract

Notes

2. Loops over dynamic variables (very low severity):

In the withdraw, `getUserDividends`, `getUserAvailable`, `getUserTotalDeposits`, and `getUserTotalWithdrawn` functions, loops unrestrictedly grow as the number of deposits increases. If you create a large number of parallel deposits from a single wallet, this can lead to an excessive increase in the transaction cost and incorrect display and processing of information.

Note: this comment is only relevant for a certain user, if he creates an excessive number of deposits (more than 300) from his wallet.

3. Closing the last payment (very low risk).

If the last user who leaves the project has a payout greater than the smart-contract balance, he will receive the entire available balance, but it will be recorded that the entire payout was closed.

Note: this comment is not critical, since after the smart contract balance is empty, it is unlikely that the contract will be used again. So it makes sense for last user to get at least something.

Independent description of the smart-contract functionality:

The Tronner contract provides the opportunity to invest any amount in TRX (from 100 TRX) in the contract and get a 200% return on investment, if the contract balance has enough funds for payment.

Dividends are payed from deposits of users (Ponzi scheme).

You can create a Deposit by calling the “invest” function and attaching the required amount of TRX to the transaction (from 100 TRX inclusive).

Each subsequent Deposit is kept separately in the contract, in order to maintain the payment amount for each Deposit.

The percentage charged to the user starts from 2% and depends on the following factors:

- For every 1,000,000 TRX on the maximum smart contract balance +0.1% until 10%. This Contract Bonus cannot decrease.
- For every 1 day of non-withdrawal of dividends from the smart contract +0.1% until 8% (when creating repeated deposits, the percent keeps growing).

Withdrawals of dividends are available at any time. Withdrawal by the use is performed by calling the “withdraw” function from the address the Deposit was made.

All dividends are calculated at the moment of request and available for withdrawal at any time.

Contract owners Commission: part of the invested funds is sent to two addresses:
(marketing address) - 9%.
(the project address) – 3%.

Three-level referral program: in the “invest” function, you can specify the address of the referrer. As a result, the referrer will get opportunity to withdraw % of the investor's Deposit according to the following table:

Referrer level	1	2	3
Percentage, %	5	2	1

Requirements for the referrer: you can not specify your own wallet as a referrer, as well as a wallet that does not have at least one contribution in the smart contract. If wrong referrer is provided, the owner (defaultReferrer) is set as referrer of the user**.**

The referrer is specified once at the time of the first deposit and is assigned to the user without the possibility of changing. From each subsequent Deposit, the referrer will get his percents.

The contract contains 16 statistical functions that do not require sending transactions:

1. `getContractBalance` – smart contract balance (with decimals, for TRX – 6 characters).
2. `getContractBalanceRate` – the current percentage for a new user.
3. `getUserPercentRate` – the current percentage for the specified user.
4. `getUserDividends` – the current amount of dividends available to withdraw.
5. `getUserCheckpoint` – the date of the last withdrawal in UNIX Time.
6. `getUserReferrer` – the user's referrer.
7. `getUserReferralBonus` – available referral bonuses for withdrawal.
8. `getUserAvailableBalanceForWithdrawal` – total available amount to withdraw (dividends + referral bonuses).
9. `isActive` – whether the user has active deposits.
10. `getUserDepositInfo` – information about the user's specified Deposit (the sequential number of the Deposit starting from 0).

11. `getUserAmountOfDeposits` – the number of user deposits.
12. `getUserTotalDeposits` – the sum of each deposits of the user.
13. `getUserTotalWithdrawn` – user dividend withdrawal amount.
14. `getUserDownlineCount` - amounts of referrals of all levels.
15. `getHoldBonus` - hold bonus of the user.
16. `getContractBonus` - contract balance bonus.

November 2020

Disclaimer

This audit is not a call to participate in the project and applies only to the Smart-Contract code at the specified address.

Do not forget that you are doing all financial actions at your own risk, especially if you deal with high-risk projects.

If you have any questions or are interested in developing/auditing of Smart-Contracts, please contact us and we will consult you.

Telegram: **@gafagilm**
E-mail: **info@grox.solutions**