THE ARCADIA GROUP

AUDIT

arcadiamgroup.com

Table of Contents

Executive Summary

A Representative Party of UTRX Team ("UTRX") engaged The Arcadia Group ("Arcadia"), a software development, research, and security company, to conduct a review of the following UTRX smart contract(s) on the UTRX repo at Commit #`356c59520561970eb098bcded3309ccbf8079018`:

1. utrx.sol

After presenting the audit to the UTRX Team, they updated the contract and solved the issues in this audit.

UTRX repo at Commit #`c86249a6dd25f2cd61745d8864c24def643f71fa`

Findings

1. Fees are not calculated proportionally

CODE-1
Severity: Medium
Impact: Medium

Contract: Utrx.sol
Category: Security
Finding Type: Dynamic
Lines: 871-883

In function depositTopMiners(), fees are not proportional for 600 USDT deposit compared to the other 2 cases, with 300 and 1500 usdt fee is 16.67% while in 600 is 25%. To get the same value you can either change the deposit value to 900 USDT instead of 600, or use only 2/3 of the fee values for a 600 USDT deposit.

```
function depositTopMiners(uint256 _amount) external returns (uint256) {
        require(
            _amount == 300000000 ||
                _amount == 600000000 ||
                _amount == 1500000000,
            "not correct amount"
        );
        usdt.safeTransferFrom(msg.sender, address(this), _amount);
//transfer from user
        uint256 availtopurchase =
            justswapexchange.getTokenToTrxInputPrice(_amount); //get full
TRX amount
        uint256 feeBig;
        uint256 feeSmall;
        uint256 feeMiddle;
        if (_amount == 300000000) {
            feeBig = justswapexchange.getTokenToTrxInputPrice(FEE_35);
//35 usdt in trx
            feeMiddle = justswapexchange.getTokenToTrxInputPrice(FEE_10);
```

```
//10 usdt in trx
            feeSmall = justswapexchange.getTokenToTrxInputPrice(FEE_5);
//5 usdt in trx
        } else if (_amount == 600000000) {
            feeBig = justswapexchange.getTokenToTrxInputPrice(FEE_105);
//105 usdt in trx
            feeMiddle = justswapexchange.getTokenToTrxInputPrice(FEE_30);
//30 usdt in trx
            feeSmall = justswapexchange.getTokenToTrxInputPrice(FEE_15);
//15 usdt in trx
        } else if (_amount == 1500000000) {
            feeBig = justswapexchange.getTokenToTrxInputPrice(FEE_175);
//175 usdt in trx
            feeMiddle = justswapexchange.getTokenToTrxInputPrice(FEE_50);
//50 usdt in trx
            feeSmall = justswapexchange.getTokenToTrxInputPrice(FEE_25);
//25 usdt in trx
        }

        availtopurchase = availtopurchase.div(10);
        uint256 deadline = block.timestamp.add(delay);
        uint256 tokensSol =
            justswapexchange.tokenToTrxSwapInput(
                _amount,
                availtopurchase,
                deadline
            );
        feeBigAddress.transfer(feeBig); //transfer fee
        feeSmallAddress.transfer(feeSmall); //transfer fee
        feeMiddleAddress.transfer(feeMiddle); //transfer fee
        emit DepositedTopMiner(msg.sender, _amount);
        return tokensSol;
    }
```

2. Operator cannot be reassigned

CODE-2
Severity: Medium
Impact: Medium

Contract: Utrx.sol
Category: AccessControl
Finding Type: Dynamic
Lines: Missing

Once set in the constructor the operator cannot be changed not even by the owner. We don't know if it's a design feature or missing, but since even the owner can be updated, probably a function to update operator by owner is missing. Consider adding a function to perform the update using onlyOwner modifier.

In this specific situation is even more important because the operator is one of the only 2 accounts(owner can too) who can send the TRXs to the user after deposit.

```
modifier onlyOwner() {
        if (msg.sender == owner) _;
    }
```

3. Unused variable

CODE-3
Severity: None
Impact: None

Contract: Utrx.sol
Category: Informational
Finding Type: Dynamic
Lines: 753

This variable is declared but never used in the actual contract, since we don't know the context of the protocol, looks like it is an unused variable. If so it could be removed while if it would be used in another contract which inherits Utrx.sol then it's fine.

```
uint256 public constant depositDelay = 24 hours;
```

IMPORTANT NOTE on funds security and control:

In the logic, a user who deposits and swaps USDT for TRX won't be able to withdraw TRX from the contract independently unless the owner or the operator does so.

While the fee is directly credited to the fee addresses and paid by the user who deposited, the TRXs stay in the contract.

The architecture in this contract requires to fully trust the owner and the operator to behave correctly.

Disclaimer

While best efforts and precautions have been taken in the preparation of this document, The Arcadia Group and the Authors assume no responsibility for errors, omissions, or damages resulting from the use of the provided information. Additionally, Arcadia would like to emphasize that the use of Arcadia's services does not guarantee the security of a smart contract or set of smart contracts and does not guarantee against attacks. One audit on its own is not enough for a project to be considered secure; that categorization can only be earned through extensive peer review and battle testing over an extended period.

Document Information

Title	UTRX Audit
Client	UTRX
Date Created	Jan 20, 2021
Date Last Edited	Jan 27, 2021
Auditor(s)	Andrea Burzi
Edited by	Joel Farris
Approved by	Rasikh Morani
Contact Details	Rasikh Morani (972) 543-3886 rasikh@arcadiamgroup.com https://t.me/thearcadiagroup