Jorge Rodriguez

CYBERSEC CONTRACT AUDIT REPORT
yTheuseMain

Introduction

I've performed an extensive audit of the smart contracts in

scope, the latest version provided by 'yTheuseMain' on October 27th,

2020. The smart contracts in scope were all audited using manual testing and personal created automated tools. In the following pages you will understand all the points that were checked.

Coverage

Target Code and Revision

For this audit, we performed research, investigation, and review of the yTheuseMain contract followed by issue reporting, along with mitigation and remediation instructions outlined in this report. The following code files are considered in-scope for the review:

yTheuseMain.sol (1036 lines).

Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

We always recommend having a bug bounty program opened to detect future bugs.

Attacks made to the contract

In order to check for the security of the contract, we tested several attacks in order to make sure that the contract is secure and follows best practices.

● Correctness of the protocol implementation;

● User funds are secure on the blockchain and cannot be transferred without user permission

● Vulnerabilities within each component as well as secure interaction between the network components

● Correctly passing requests to the network core

● Data privacy, data leaking, and information integrity

● Key management implementation: secure private key storage and proper management of encryption and signing keys

● Handling large volumes of network traffic

● Resistance to DDoS and similar attacks

● Aligning incentives with the rest of the network

● Any attack that impacts funds, such as draining or manipulating of funds

● Mismanagement of funds via transactions

● Inappropriate permissions and excess authority

● Special token issuance model

Over and under flows

An overflow happens when the limit of the type variable uint256, 2 ** 256, is exceeded. What happens is that the value resets to zero instead of incrementing more. On the other hand, an underflow happens when you try to subtract 0 minus a number bigger than 0. For example, if you subtract 0 - 1 the result will be = 2 ** 256 instead of -1. This is quite dangerous.

Safemath library is used in the contract by using OpenZeppelin's SafeMath mitigating that attack vector. Contract is **not vulnerable**.

Short address attack

If the token contract has enough amount of tokens and the 'buy' function doesn't check the length of the address of the sender, the Tron's virtual machine will just add zeros to the transaction until the address is complete.

Although this contract is not vulnerable to this attack, there are some points where users can mess themselves up due to this (**Please see below**). It is highly recommended to call functions after checking the validity of the address.

Visibility & Delegate call

It is also known as, The Parity Hack, which occurs while misuse of Delegate call.

No such issues found in this smart contract and visibility also properly addressed.

The contract is **not prone to any vulnerability** due to this in this case.

Reentrancy / TheDAO hack

Reentrancy occurs in this case: any interaction from a contract (A) with another contract (B) and any transfer of Tron hands over control to that contract (B). This makes it possible for B to call back into A before this interaction is completed.

Use of "require" function in this smart contract mitigated this vulnerability.

Forcing Tron to a contract

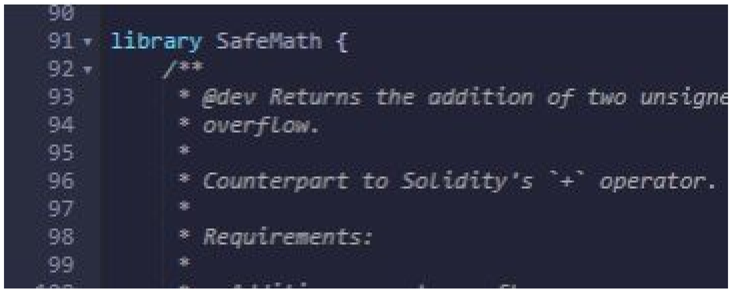
While implementing "selfdestruct" in smart contract, it sends all the tron to the target address. Now, if the target address is a contract address, then the fallback function of target contract does not get called. And thus Hacker can bypass the "Required" conditions.

Here, the Smart Contract's balance has never been used as guard, which mitigated this vulnerability.

Reviews [OK] from the contract

1. **Safe Math Library**

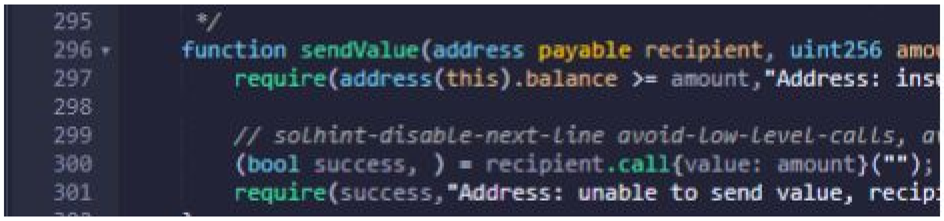
SafeMath is used to avoid buffer/underflow attacks.



```
library SafeMath {
    /**
     * @dev Returns the addition of two unsigne
     * overflow.
     *
     * Counterpart to Solidity's `+` operator.
     *
     * Requirements:
     *
```

2.**Require condition in functions**

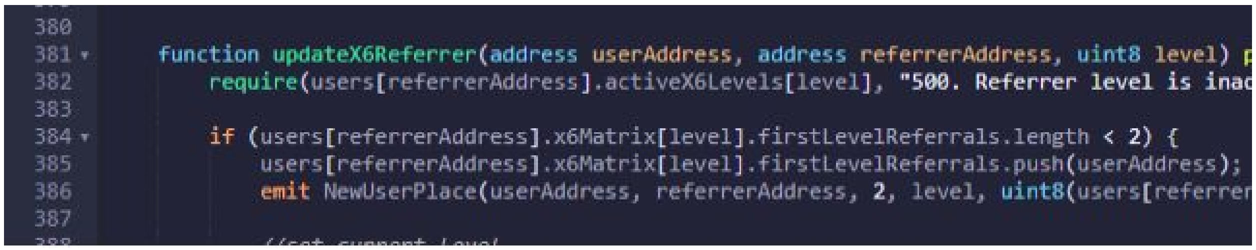
Here you are checking that the amount value is less or equal to the contract balance and, checking that successfully transfer amount to the recipient address.



```
    */
    function sendValue(address payable recipient, uint256 amo
        require(address(this).balance >= amount,"Address: ins

        // solhint-disable-next-line avoid-low-level-calls, a
        (bool success, ) = recipient.call{value: amount}("");
        require(success,"Address: unable to send value, recip
```

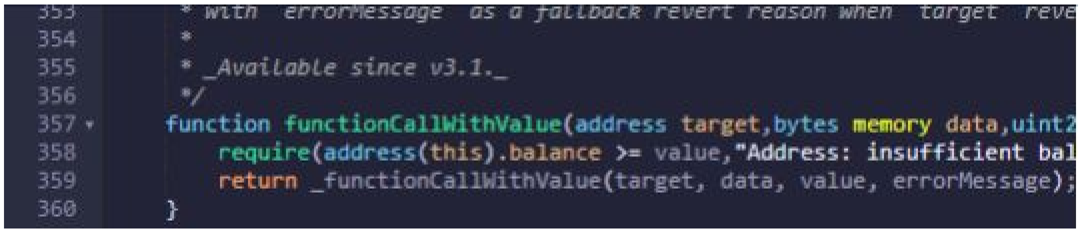
3.

Here you are checking if that Referrer level is active or not.



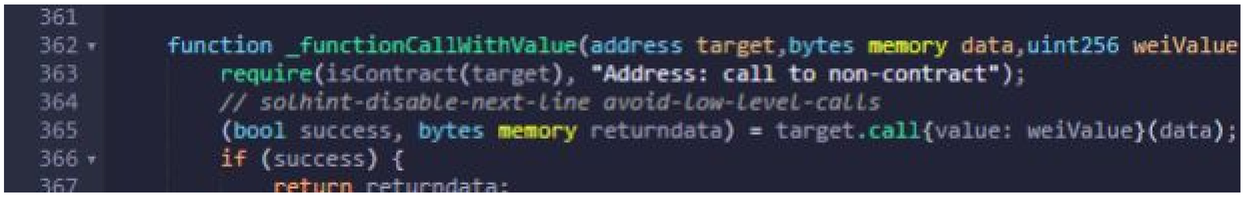
```
    function updateX6Referrer(address userAddress, address referrerAddress, uint8 level) p
        require(users[referrerAddress].activeX6Levels[level], "500. Referrer level is inac

        if (users[referrerAddress].x6Matrix[level].firstLevelReferrals.length < 2) {
            users[referrerAddress].x6Matrix[level].firstLevelReferrals.push(userAddress);
            emit NewUserPlace(userAddress, referrerAddress, 2, level, uint8(users[referrer
```

Here you are checking that value is less or equal to contract balance.



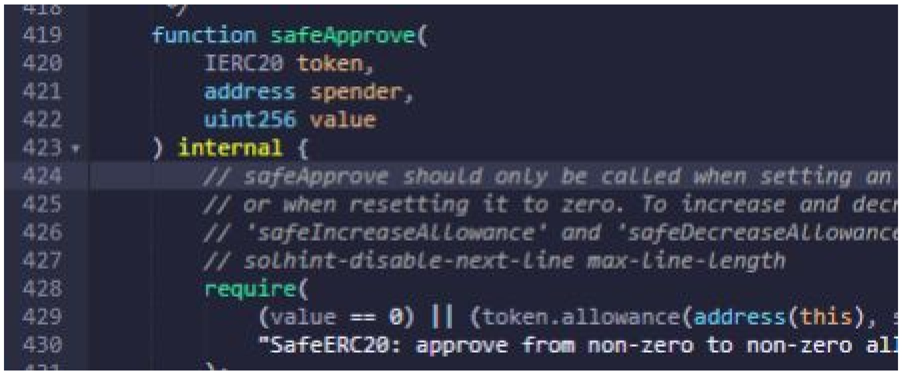
```
     * with  errorMessage  as a fallback revert reason when  target  reve
     *
     * _Available since v3.1._
     */
    function functionCallWithValue(address target,bytes memory data,uint2
        require(address(this).balance >= value,"Address: insufficient bal
        return _functionCallWithValue(target, data, value, errorMessage);
    }
```

Here you are checking that if the target address is a Contract or not.



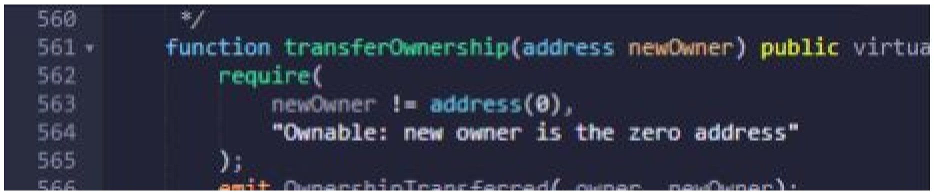
```

    function _functionCallWithValue(address target,bytes memory data,uint256 weiValue
        require(isContract(target), "Address: call to non-contract");
        // solhint-disable-next-line avoid-low-level-calls
        (bool success, bytes memory returndata) = target.call{value: weiValue}(data);
        if (success) {
            return returndata;
```

Here you are checking that allowance function will not call with 0 value.



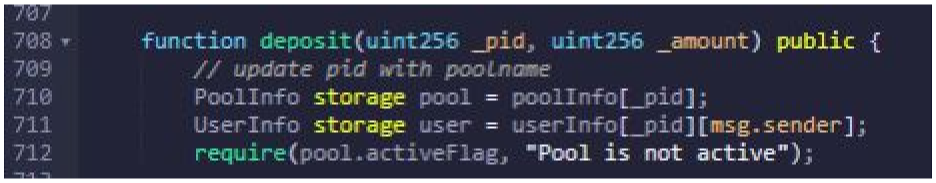
```
     */
    function safeApprove(
        IERC20 token,
        address spender,
        uint256 value
    ) internal {
        // safeApprove should only be called when setting an
        // or when resetting it to zero. To increase and decr
        // 'safeIncreaseAllowance' and 'safeDecreaseAllowance
        // solhint-disable-next-line max-line-length
        require(
            (value == 0) || (token.allowance(address(this), s
            "SafeERC20: approve from non-zero to non-zero all
        );
```

Here you are checking that newOwner address value is properly set up



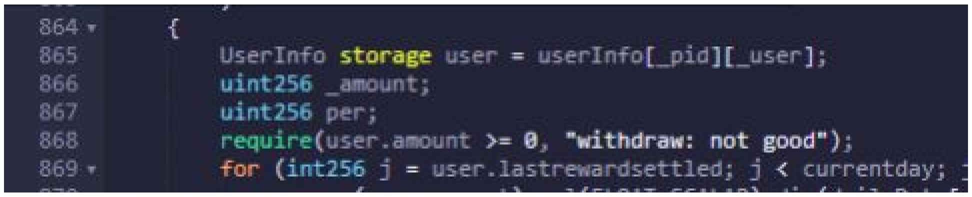
```
     */
    function transferOwnership(address newOwner) public virtua
        require(
            newOwner != address(0),
            "Ownable: new owner is the zero address"
        );
        emit OwnershipTransferred(_owner, newOwner);
```

Here you are checking that pool is active for deposit.



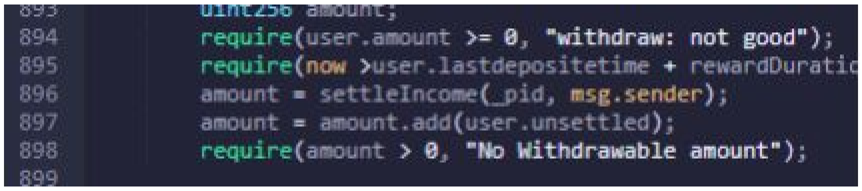
```
    function deposit(uint256 _pid, uint256 _amount) public {
        // update pid with poolname
        PoolInfo storage pool = poolInfo[_pid];
        UserInfo storage user = userInfo[_pid][msg.sender];
        require(pool.activeFlag, "Pool is not active");
```

Here you are checking that user.amount is bigger than 0.



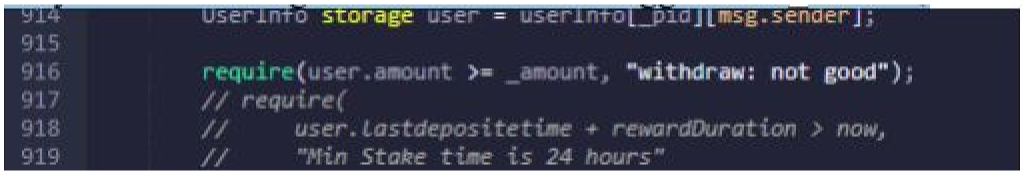
```
    {
        UserInfo storage user = userInfo[_pid][_user];
        uint256 _amount;
        uint256 per;
        require(user.amount >= 0, "withdraw: not good");
        for (int256 j = user.lastrewardsettled; j < currentday;
```

Here you are checking that the user.amount is bigger than 0, you can withdraw money after 24 hours of deposit and, if amount comes 0 then you cannot call withdraw.



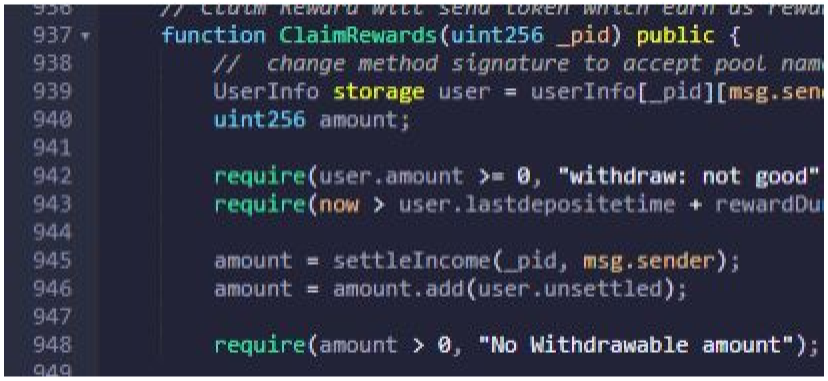
```
        uint256 amount;
        require(user.amount >= 0, "withdraw: not good");
        require(now >user.lastdepositetime + rewardDuratic
        amount = settleIncome(_pid, msg.sender);
        amount = amount.add(user.unsettled);
        require(amount > 0, "No Withdrawable amount");
```

Here you are checking that user.amount is bigger than _amount.



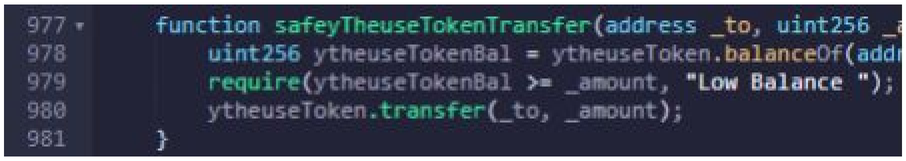
```
        UserInfo storage user = userInfo[_pid][msg.sender];

        require(user.amount >= _amount, "withdraw: not good");
        // require(
        //     user.lastdepositetime + rewardDuration > now,
        //     "Min Stake time is 24 hours"
```

Here you are checking that the user.amount is bigger than 0, you can withdraw money after 24 hours of deposit and, if amount comes 0 then you cannot call withdraw.



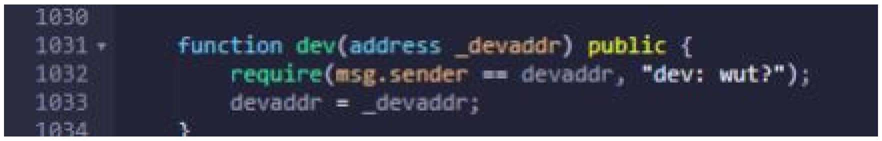
```
    // Claim Reward will send token which earn us rewa
    function ClaimRewards(uint256 _pid) public {
        //  change method signature to accept pool nam
        UserInfo storage user = userInfo[_pid][msg.sen
        uint256 amount;

        require(user.amount >= 0, "withdraw: not good"
        require(now > user.lastdepositetime + rewardDu

        amount = settleIncome(_pid, msg.sender);
        amount = amount.add(user.unsettled);

        require(amount > 0, "No Withdrawable amount");
```

Here you are checking that ytheuserTokenBal is bigger or equal to _amount.



```
    function safeyTheuseTokenTransfer(address _to, uint256 _a
        uint256 ytheuseTokenBal = ytheuseToken.balanceOf(addr
        require(ytheuseTokenBal >= _amount, "Low Balance ");
        ytheuseToken.transfer(_to, _amount);
    }
```

Here you are checking that msg.sender is not _devaddr.



```

    function dev(address _devaddr) public {
        require(msg.sender == devaddr, "dev: wut?");
        devaddr = _devaddr;
    }
```

Critical / Medium Vulnerabilities found

No critical / medium vulnerabilities found.

Low Vulnerabilities found

Compiler version

In the contract you used “pragma solidity 0.6.12” to define the compiler version.

Solidity source files indicate the versions of the compiler that can be used in the contract.

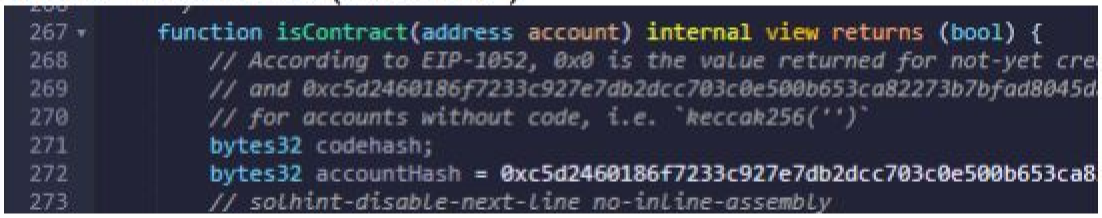
Pragma solidity >=0.6.12 will allow you to compile version 0.6.12 and above. If you don't use >= you will be able to compile the 0.6.12 version only. If there are major changes in the compiler version and get outdated you may have issues in a future.

Short Address Attack

This is not a big issue in solidity, because nowadays it is increased In the new solidity version. But it is good practice to Check for the short address. => After updating the version of solidity it’s not mandatory. => In some functions you are not checking the value of Address parameter

This is not an issue in solidity anymore but nowadays new solidity versions are having some issues regarding it. Is a good practice to check short addresses. In some functions you aren’t checking the value of the Address parameter.

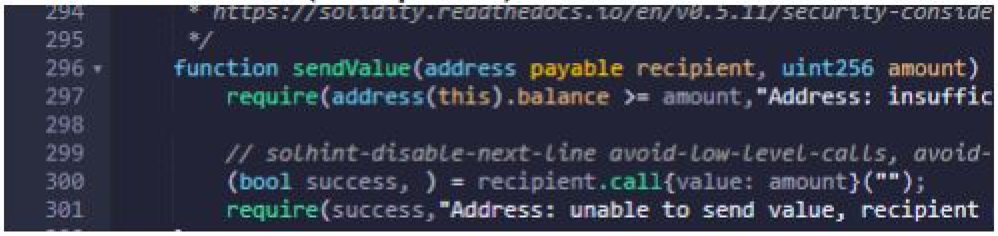
Function: - isContract (‘account’)



```
    function isContract(address account) internal view returns (bool) {
        // According to EIP-1052, 0x0 is the value returned for not-yet cre
        // and 0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d
        // for accounts without code, i.e. `keccak256('')`
        bytes32 codehash;
        bytes32 accountHash = 0xc5d2460186f7233c927e7db2dcc703c0e500b653ca8
        // solhint-disable-next-line no-inline-assembly
```

It's necessary to check the address value of "account". Because here you are passing whatever variable comes in the "account" address from outside.

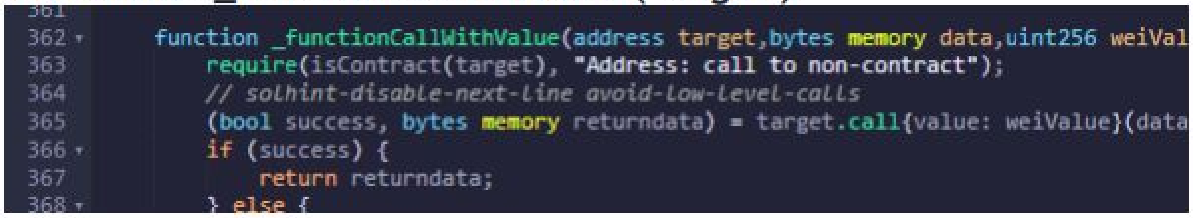
Function: - sendValue (‘recipient’)



```
     * https://solidity.readthedocs.io/en/v0.5.11/security-conside
     */
    function sendValue(address payable recipient, uint256 amount)
        require(address(this).balance >= amount,"Address: insuffic

        // solhint-disable-next-line avoid-low-level-calls, avoid-
        (bool success, ) = recipient.call{value: amount}("");
        require(success,"Address: unable to send value, recipient
```

It's necessary to check the address value of "recipient". Because here you are passing whatever variable comes in "recipient" address from outside.

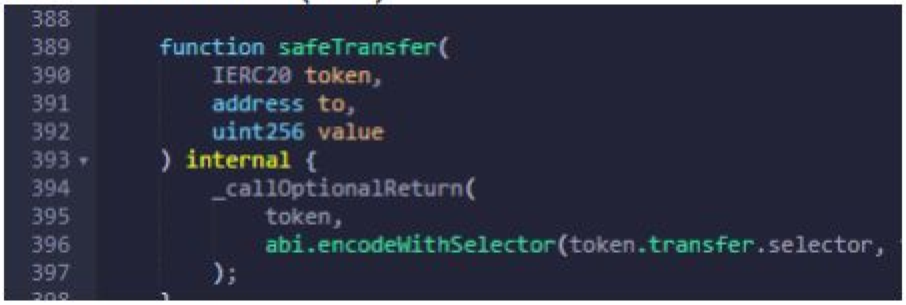
Function: - _functionCallWithValue (‘target’)



```
    function _functionCallWithValue(address target,bytes memory data,uint256 weiVal
        require(isContract(target), "Address: call to non-contract");
        // solhint-disable-next-line avoid-low-level-calls
        (bool success, bytes memory returndata) = target.call{value: weiValue}(data
        if (success) {
            return returndata;
        } else {
```

It's necessary to check the address value of "target". Because here you are passing whatever variable comes in the "target" address from outside.

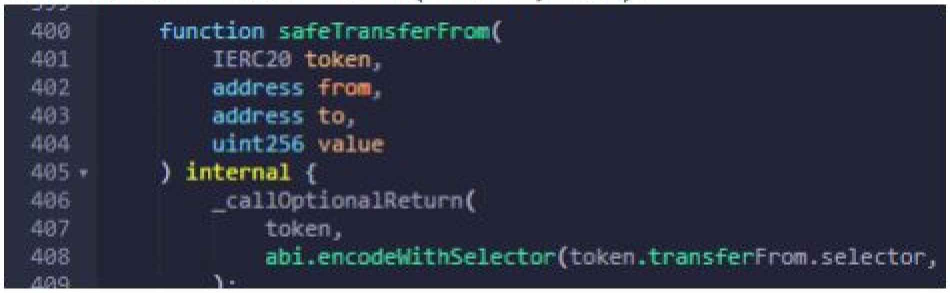
Function: - safeTransfer (‘to’)



```

    function safeTransfer(
        IERC20 token,
        address to,
        uint256 value
    ) internal {
        _callOptionalReturn(
            token,
            abi.encodeWithSelector(token.transfer.selector,
        );
```

It's necessary to check the address value of "to". Because here you are passing whatever variable comes in "to" address from outside.

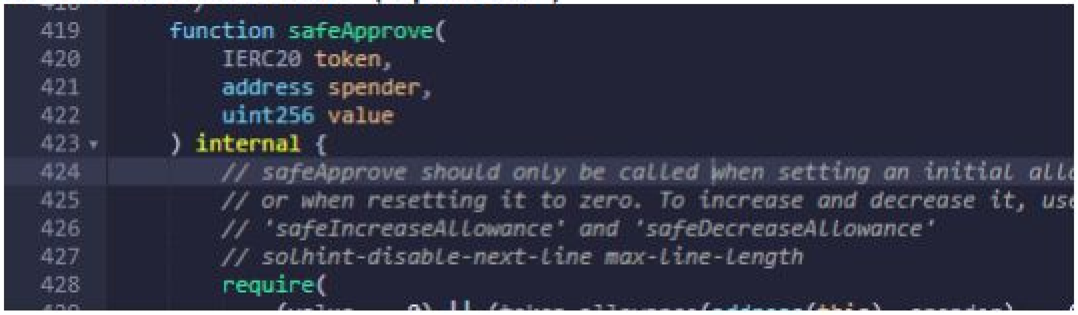
Function: - safeTransferFrom ('from', 'to')



```
    function safeTransferFrom(
        IERC20 token,
        address from,
        address to,
        uint256 value
    ) internal {
        _callOptionalReturn(
            token,
            abi.encodeWithSelector(token.transferFrom.selector,
```

It's necessary to check the addresses value of "from","to". Because here you are passing whatever variable comes in "from","to" addresses from outside.

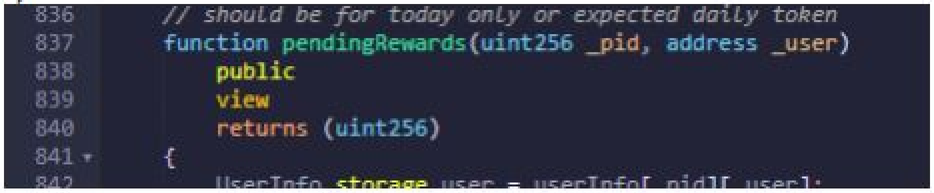
Function: - safeApprove, safeIncreaseAllowance, safeDecreaseAllowance ('spender')



```
    function safeApprove(
        IERC20 token,
        address spender,
        uint256 value
    ) internal {
        // safeApprove should only be called when setting an initial all
        // or when resetting it to zero. To increase and decrease it, use
        // 'safeIncreaseAllowance' and 'safeDecreaseAllowance'
        // solhint-disable-next-line max-line-length
        require(
```

It's necessary to check the address value of "spender". Because here you are passing whatever variable comes in the "spender" address from outside.

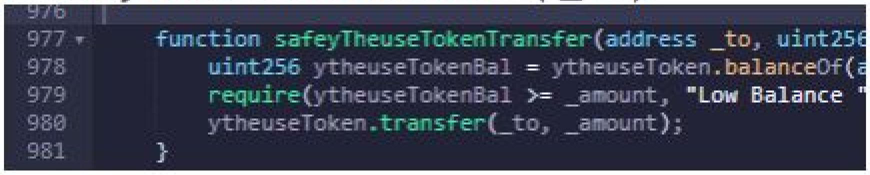
Function: - pendingRewards, settleIncome, CurrentShareofPool ('_user')



```
    // should be for today only or expected daily token
    function pendingRewards(uint256 _pid, address _user)
        public
        view
        returns (uint256)
    {
```

It's necessary to check the address value of "_user". Because here you are passing whatever variable comes in "_user" address from outside.

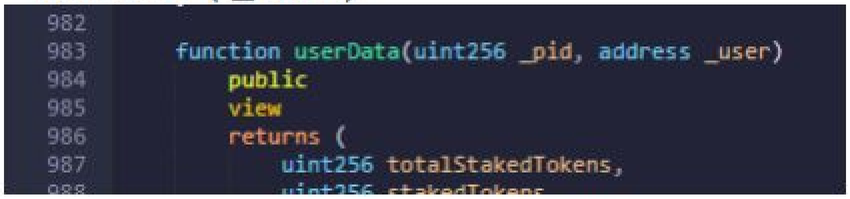
Function: - safeyTheuseTokenTransfer ('_to')



```
    function safeyTheuseTokenTransfer(address _to, uint256
        uint256 ytheuseTokenBal = ytheuseToken.balanceOf(a
        require(ytheuseTokenBal >= _amount, "Low Balance "
        ytheuseToken.transfer(_to, _amount);
    }
```

It's necessary to check the address value of "_to". Because here you are passing whatever variable comes in "_to" address from outside.

Function: - userDate ('_user')



```
    function userData(uint256 _pid, address _user)
        public
        view
        returns (
            uint256 totalStakedTokens,
```

It's necessary to check the address value of "_user". Because here you are passing whatever variable comes in the "_user" address from outside.

Unchecked return value or response

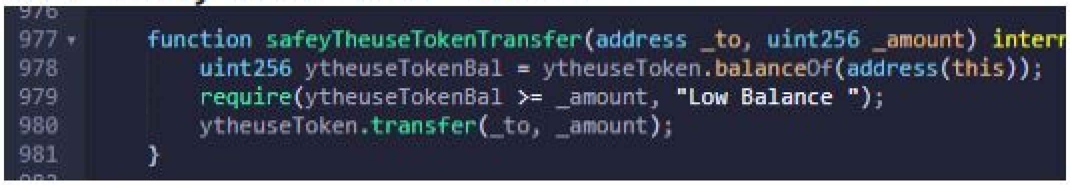
You are transferring funds to address using a transfer method.

It is always good to check the return value or response from a function call.

Here are some functions where you forgot to check a response.

I suggest, if there is a possibility then please check the response.

Function: - safeyTheuseTokenTransfer



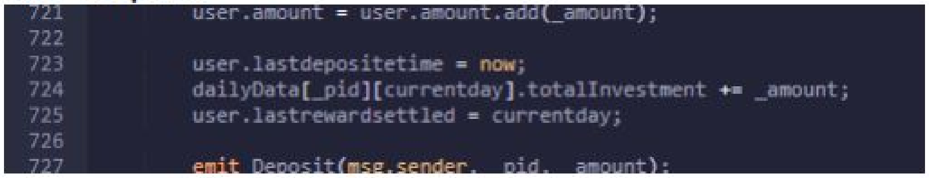
```
    function safeyTheuseTokenTransfer(address _to, uint256 _amount) intern
        uint256 ytheuseTokenBal = ytheuseToken.balanceOf(address(this));
        require(ytheuseTokenBal >= _amount, "Low Balance ");
        ytheuseToken.transfer(_to, _amount);
    }
```

Here you are calling the transfer method 1 time. It is good to check that the transfer is successfully done or not.

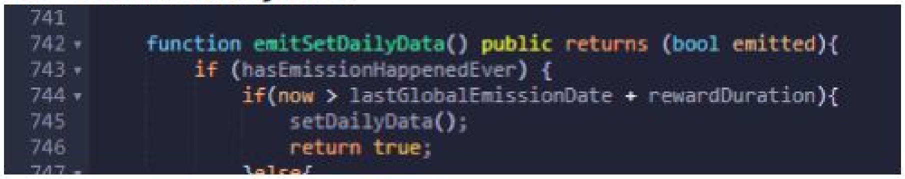
Safemath implemented in the next functions

You have implemented safemath in the contract but there are few functions that aren't using it, **please review it**.

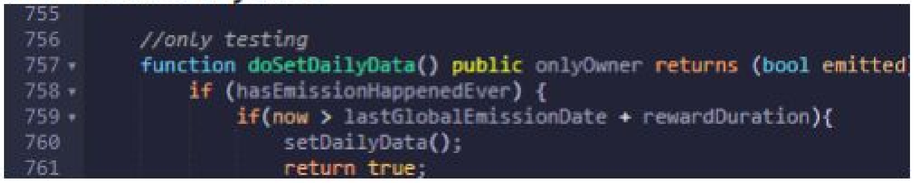
Function: - deposit


```
        user.amount = user.amount.add(_amount);

        user.lastdeposittime = now;
        dailyData[_pid][currentday].totalInvestment += _amount;
        user.lastrewardsettled = currentday;

        emit Deposit(msg.sender, _pid, _amount);
```

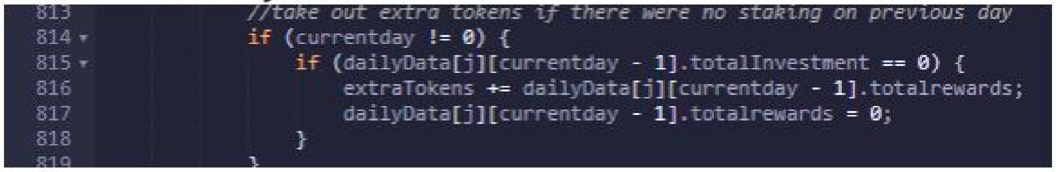
Function: - emitSetDailyData


```

    function emitSetDailyData() public returns (bool emitted){
        if (hasEmissionHappenedEver) {
            if(now > lastGlobalEmissionDate + rewardDuration){
                setDailyData();
                return true;
            }else{
```

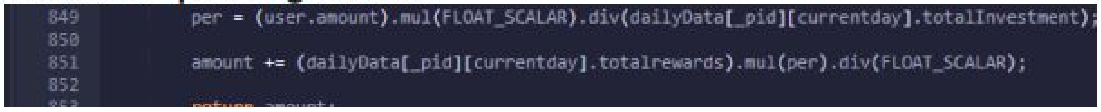
Function: - doSetDailyData


```

    //only testing
    function doSetDailyData() public onlyOwner returns (bool emitted
        if (hasEmissionHappenedEver) {
            if(now > lastGlobalEmissionDate + rewardDuration){
                setDailyData();
                return true;
```

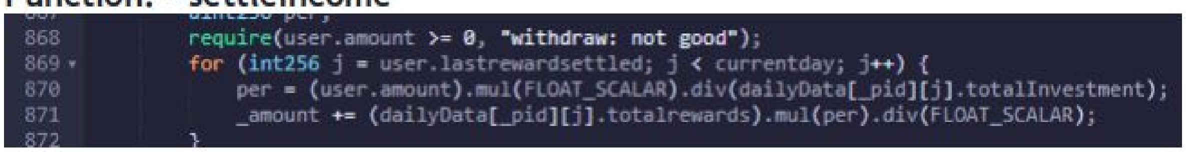
Function: - setDailyData


```
            //take out extra tokens if there were no staking on previous day
            if (currentday != 0) {
                if (dailyData[j][currentday - 1].totalInvestment == 0) {
                    extraTokens += dailyData[j][currentday - 1].totalrewards;
                    dailyData[j][currentday - 1].totalrewards = 0;
                }
            }
```

Function: - pendingRewards


```
        per = (user.amount).mul(FLOAT_SCALAR).div(dailyData[_pid][currentday].totalInvestment);

        amount += (dailyData[_pid][currentday].totalrewards).mul(per).div(FLOAT_SCALAR);

        return amount;
```

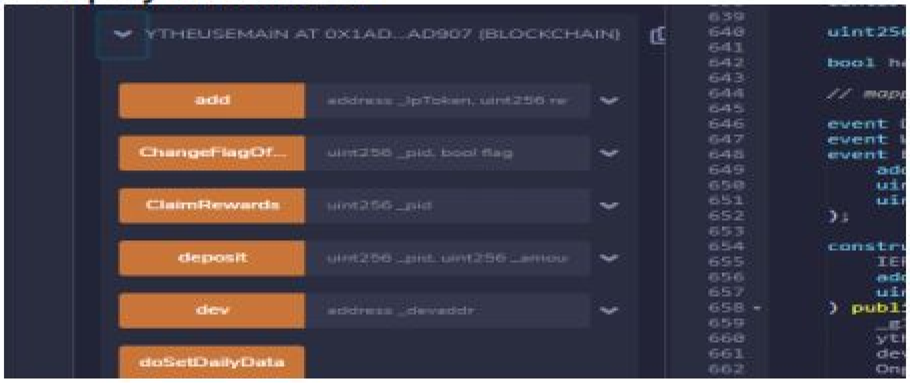
Function: - settleIncome


```
        require(user.amount >= 0, "withdraw: not good");
        for (int256 j = user.lastrewardsettled; j < currentday; j++) {
            per = (user.amount).mul(FLOAT_SCALAR).div(dailyData[_pid][j].totalInvestment);
            _amount += (dailyData[_pid][j].totalrewards).mul(per).div(FLOAT_SCALAR);
        }
```

Deployment on Testnet

POC (Proof of concept):

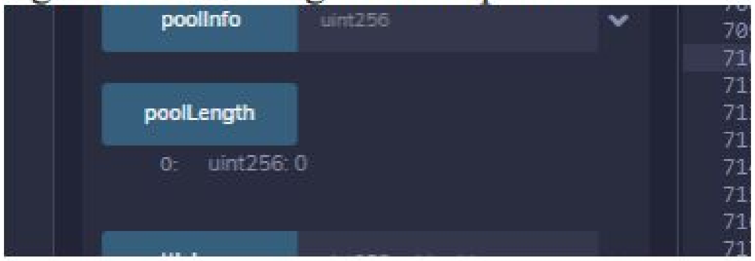
Function: - Deploy on testnet:-



https://goerli.etherscan.io/tx/0x42558c477862ada3f3cf618899e9ed073f2f68ec109d3da99c43421df3d076f3

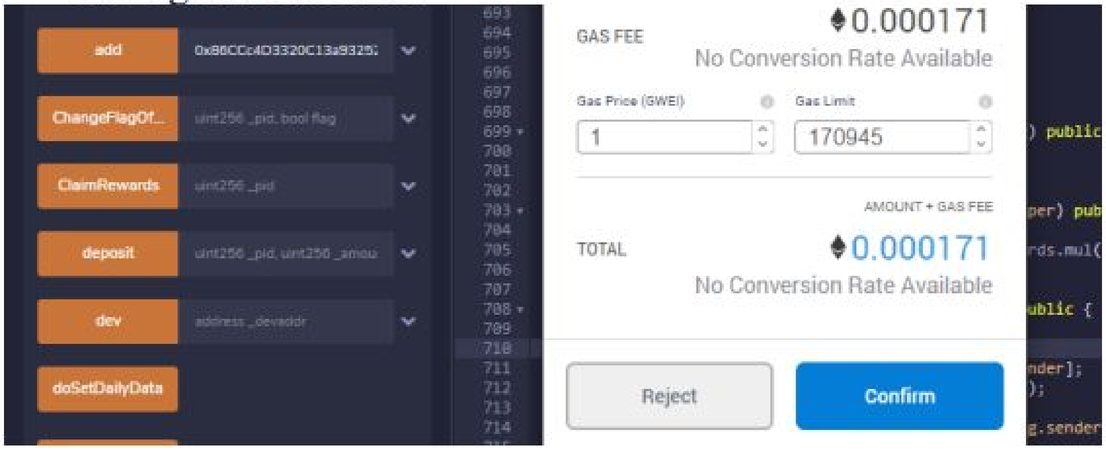
Function: - Admin has to add token in pool:-

- PoolLength before adding token to pool.



0: uint256: 0

- Calling add function.



add: 0x86CCc4D3320C13a93251

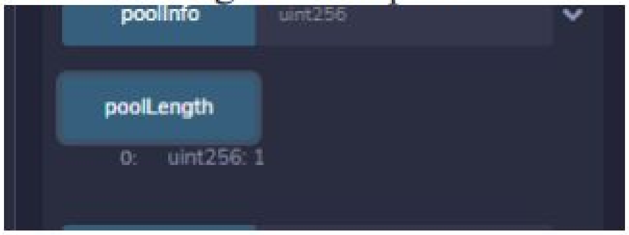
GAS FEE ♦0.000171
No Conversion Rate Available

Gas Price (GWEI): 1
Gas Limit: 170945

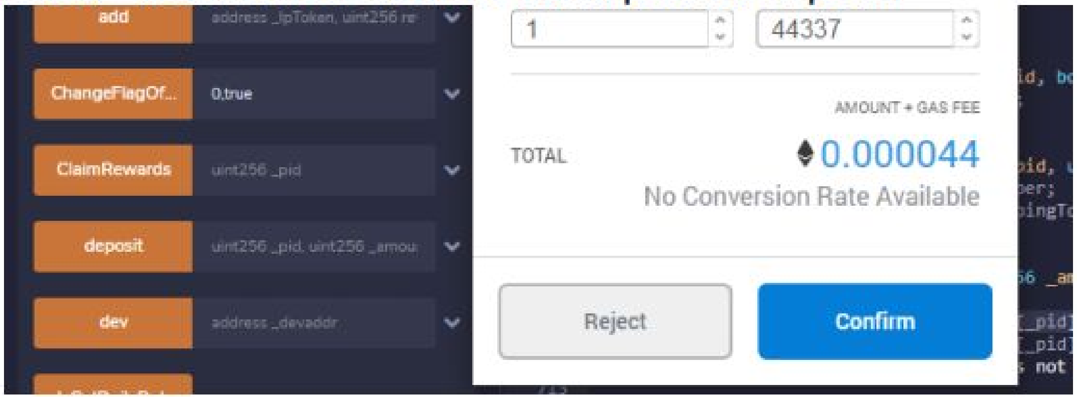
AMOUNT + GAS FEE
TOTAL ♦0.000171
No Conversion Rate Available

https://goerli.etherscan.io/tx/0xc49974363a249c959ffa5dab233e7187ba356d703b09f9430f2548ec35a55700

- PoolLength after adding token to pool.



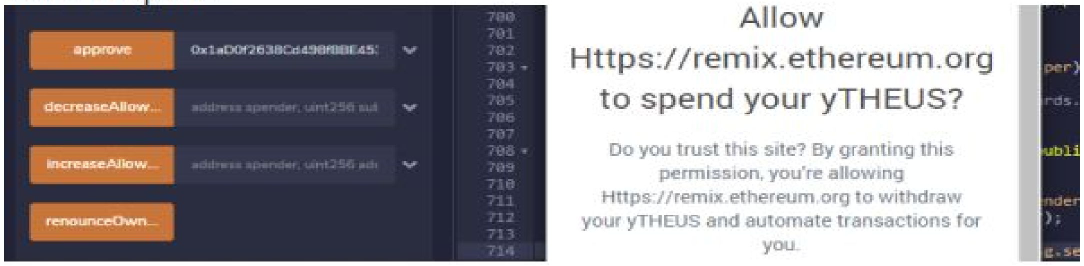
Function: - Admin has to activate token pool for deposit:-



https://goerli.etherscan.io/tx/0x8dc83d29065a47cc7cdab333e1f4d30e3227765199afcf23

Function: - deposit function call:-

- First need to approve this contract in token contract with the amount you want to deposit.

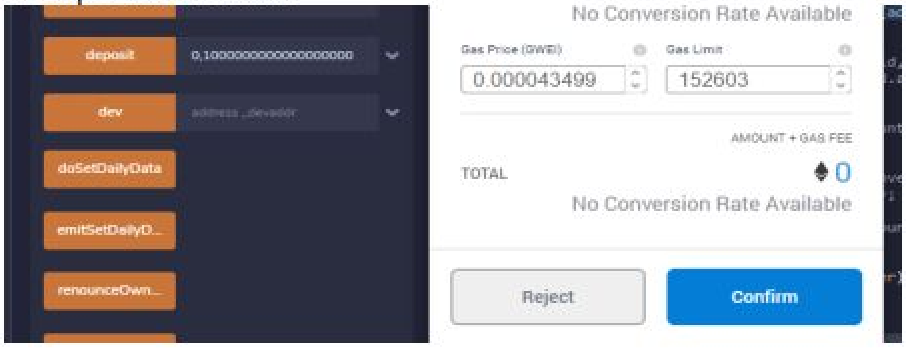


Allow Https://remix.ethereum.org to spend your yTHEUS?

Do you trust this site? By granting this permission, you're allowing Https://remix.ethereum.org to withdraw your yTHEUS and automate transactions for you.

https://goerli.etherscan.io/tx/0x834d599ee2e28a82be6b3c105a442c9f725914aaf0523ec08f429f41f1acf452

- Now call deposit function.



No Conversion Rate Available

Gas Price (GWEI) 0.000043499

Gas Limit 152603

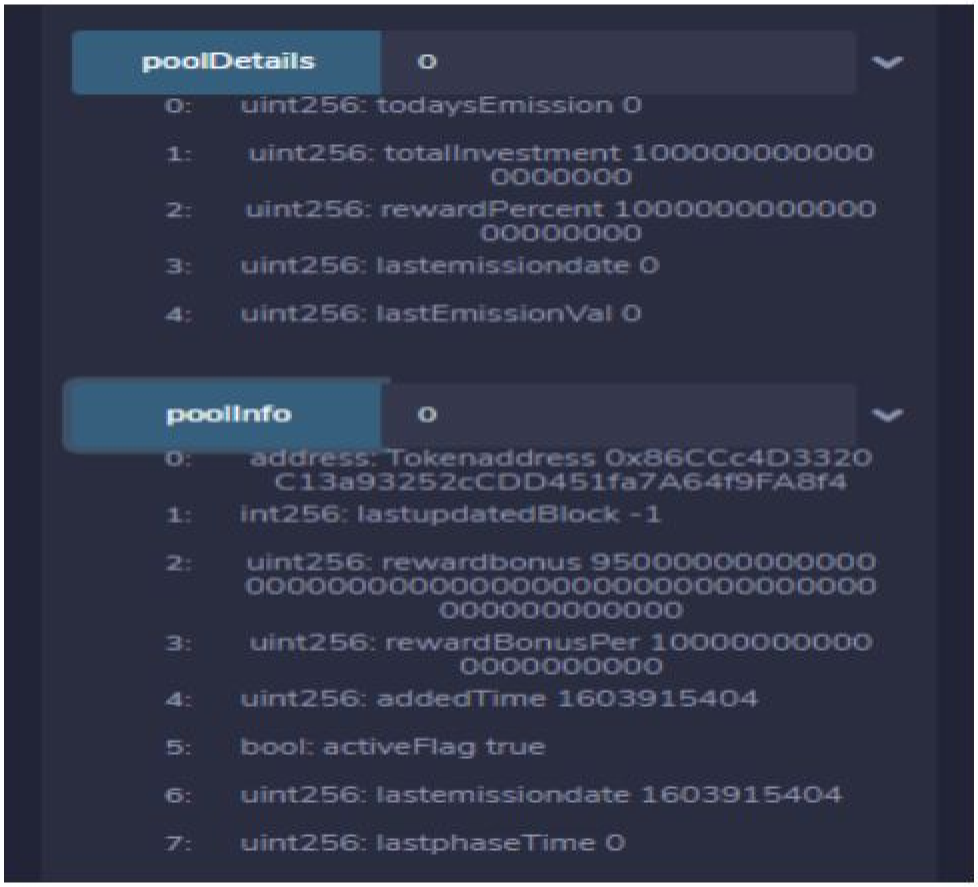
AMOUNT + GAS FEE

TOTAL 0

No Conversion Rate Available

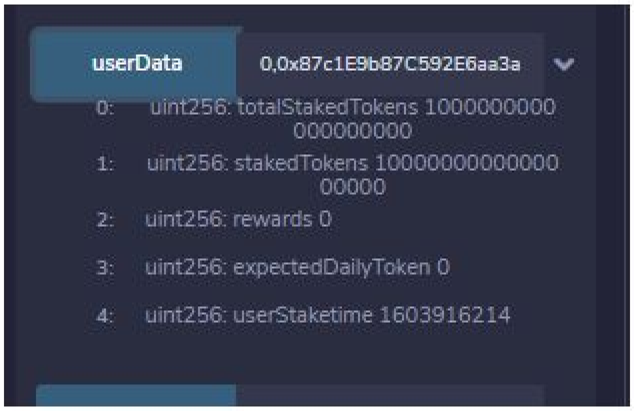
https://goerli.etherscan.io/tx/0xefffbf6ed54a3b144c94451f9e2c8d668fcaf220315f0c74d1fe70c660511d8e

Function: - Pool information with token deposit:-



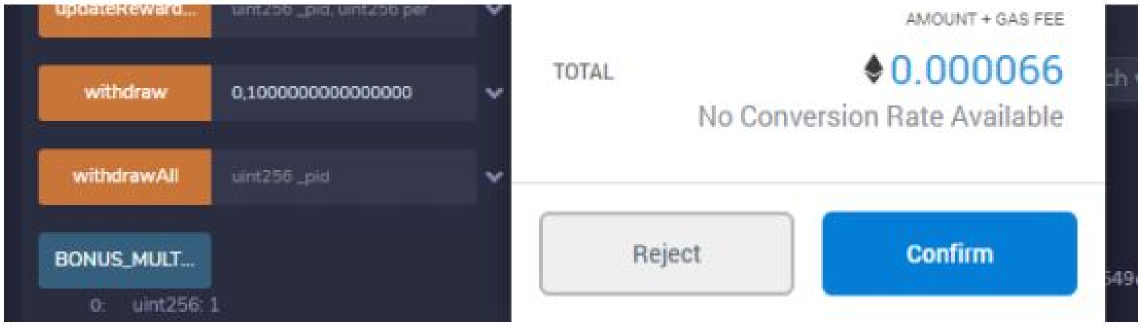
```
poolDetails    0
0:  uint256: todaysEmission 0
1:  uint256: totalInvestment 1000000000000000000
2:  uint256: rewardPercent 100000000000000000000
3:  uint256: lastemissiondate 0
4:  uint256: lastEmissionVal 0

poolInfo    0
0:  address: Tokenaddress 0x86CCc4D3320C13a93252cCDD451fa7A64f9FA8f4
1:  int256: lastupdatedBlock -1
2:  uint256: rewardbonus 95000000000000000000000000000000000000000000000000000000
3:  uint256: rewardBonusPer 100000000000000000000
4:  uint256: addedTime 1603915404
5:  bool: activeFlag true
6:  uint256: lastemissiondate 1603915404
7:  uint256: lastphaseTime 0
```

Function: - User data with totalStakedToken:-



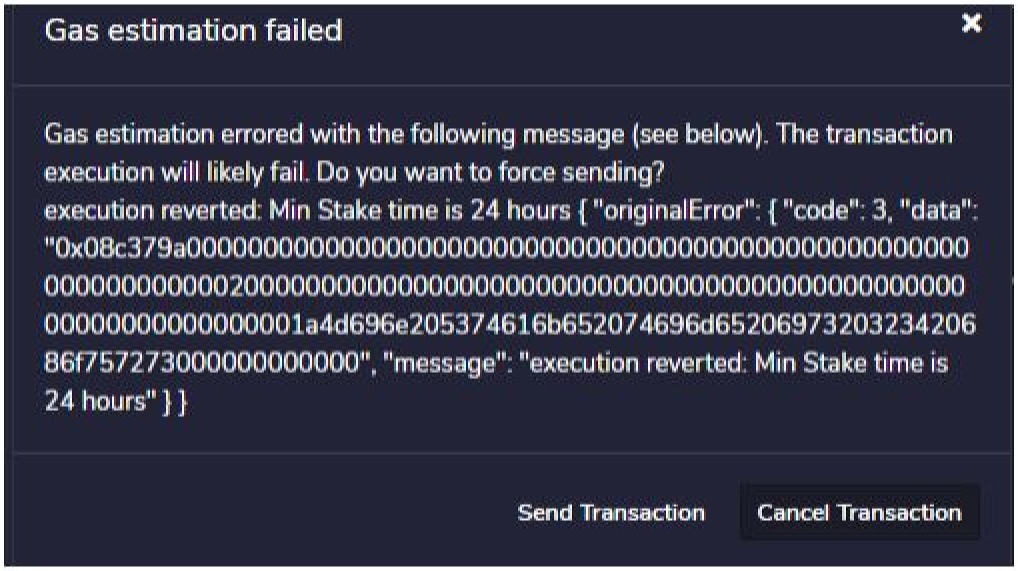
```
userData    0,0x87c1E9b87C592E6aa3a
0:  uint256: totalStakedTokens 1000000000000000000
1:  uint256: stakedTokens 100000000000000000
2:  uint256: rewards 0
3:  uint256: expectedDailyToken 0
4:  uint256: userStaketime 1603916214
```

Function: - Withdraw function:-



https://goerli.etherscan.io/tx/0xe47323d2d060af0368c13a28748b36df3b783f9eb89ba549ea9cab5bfcba485b

Function: - WithdrawAll function:-


Gas estimation failed

Gas estimation errored with the following message (see below). The transaction execution will likely fail. Do you want to force sending?
execution reverted: Min Stake time is 24 hours { "originalError": { "code": 3, "data": "0x08c379a0000000000000000000000000000000000000000000000000000000000000000000200000000000000000000000000000000000000000000000000000000000000000001a4d696e205374616b652074696d6520697320323420686f757273000000000000", "message": "execution reverted: Min Stake time is 24 hours" } }

- I can call withdraw after 24 hours of deposit. So this error is correct.

Summary of the Audit

Overall the code is good and performs well.

Please try to check the address and value of the token externally before sending to the solidity code and also review the code version of solidity.

• Note: Please focus on a version, check the response of the transfer method, use safemath library in some methods that are not used, and check addresses.

I have seen that a developer is using this method now, so I like to tell you that writing smart contracts with the notion that block values are not precise, and the use of them **can lead to unexpected effects**.