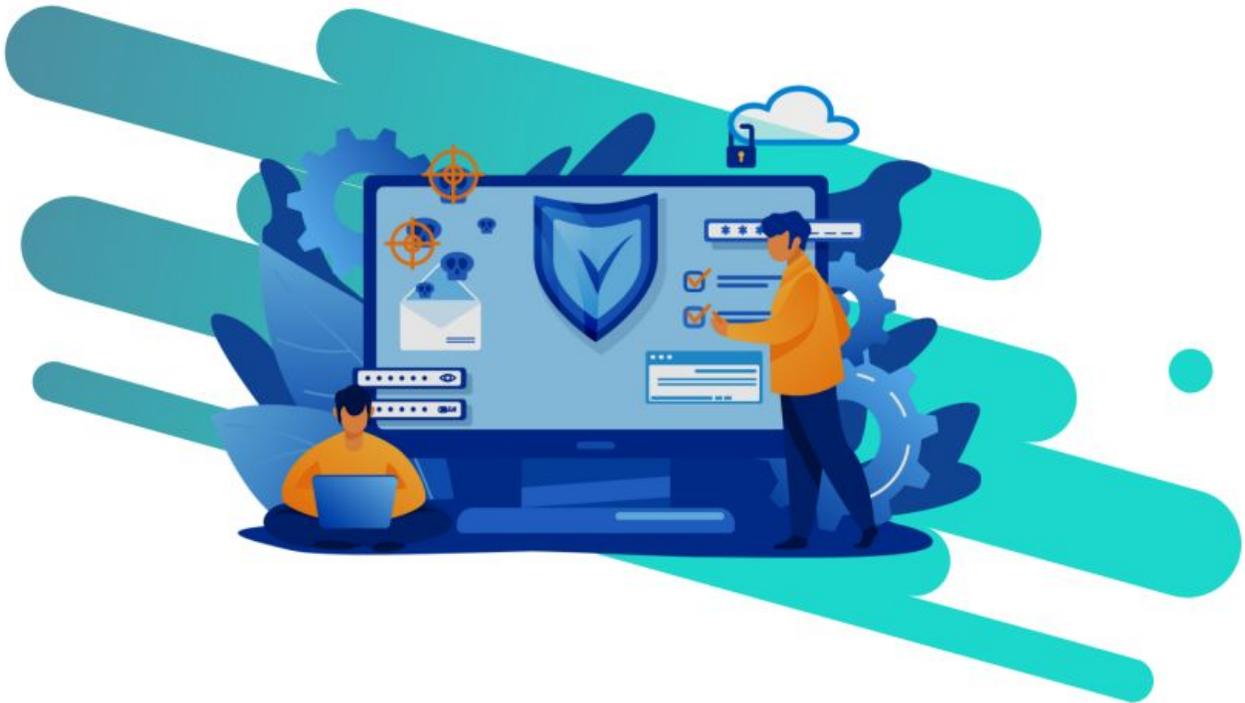


Jorge Rodriguez (<https://www.linkedin.com/in/jorge-rodriguez-115570139>)  
<https://www.ctdsec.com>

# 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 Ethereum 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 Ethereum 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 Eth to a contract

While implementing "selfdestruct" in smart contract, it sends all the eth 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.

```
90
91 ▾ library SafeMath {
92 ▾     /**
93      * @dev Returns the addition of two unsigned
94      * overflow.
95      *
96      * Counterpart to Solidity's `+` operator.
97      *
98      * Requirements:
99      *
100     */
```

### 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.

```
295     */
296 ▾     function sendValue(address payable recipient, uint256 amount) public {
297         require(address(this).balance >= amount, "Address: insufficient balance");
298
299         // solhint-disable-next-line avoid-low-level-calls, avoid-unsafe-blocks
300         (bool success, ) = recipient.call{value: amount}("");
301         require(success, "Address: unable to send value, recipient may have died");
302     }
```

3.

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

```
380
381 ▾     function updateX6Referrer(address userAddress, address referrerAddress, uint8 level) public {
382         require(users[referrerAddress].activeX6Levels[level], "500. Referrer level is inactive");
383
384         if (users[referrerAddress].x6Matrix[level].firstLevelReferrals.length < 2) {
385             users[referrerAddress].x6Matrix[level].firstLevelReferrals.push(userAddress);
386             emit NewUserPlace(userAddress, referrerAddress, 2, level, uint8(users[referrerAddress].activeX6Levels[level]));
387         }
388     }
```

---

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

```
353 * with errorMessage as a fallback revert reason when target reverts
354 *
355 * _Available since v3.1._
356 */
357 function functionCallWithValue(address target, bytes memory data, uint256
358     value) public virtual {
359     require(address(this).balance >= value, "Address: insufficient balance");
360     return _functionCallWithValue(target, data, value, errorMessage);
361 }
```

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

```
361
362 function _functionCallWithValue(address target, bytes memory data, uint256 weiValue
363     value) private {
364     require(isContract(target), "Address: call to non-contract");
365     // solhint-disable-next-line avoid-low-level-calls
366     (bool success, bytes memory returndata) = target.call{value: weiValue}(data);
367     if (success) {
368         return returndata;
369     }
370 }
```

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

```
418
419 function safeApprove(
420     IERC20 token,
421     address spender,
422     uint256 value
423 ) internal {
424     // safeApprove should only be called when setting an allowance for a
425     // spender or when resetting it to zero. To increase and decrease
426     // allowance, use 'safeIncreaseAllowance' and 'safeDecreaseAllowance'
427     // solhint-disable-next-line max-line-length
428     require(
429         (value == 0) || (token.allowance(address(this), spender) > 0),
430         "SafeERC20: approve from non-zero to non-zero allowance"
431     );
432     token.approve(spender, value);
433 }
```

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

```
560
561 function transferOwnership(address newOwner) public virtual {
562     require(
563         newOwner != address(0),
564         "Ownable: new owner is the zero address"
565     );
566     emit OwnershipTransferred(owner, newOwner);
567 }
```

Here you are checking that pool is active for deposit.

---

```

707
708 ▾ function deposit(uint256 _pid, uint256 _amount) public {
709     // update pid with poolname
710     PoolInfo storage pool = poolInfo[_pid];
711     UserInfo storage user = userInfo[_pid][msg.sender];
712     require(pool.activeFlag, "Pool is not active");
713

```

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

```

864 ▾ {
865     UserInfo storage user = userInfo[_pid][_user];
866     uint256 _amount;
867     uint256 per;
868     require(user.amount >= 0, "withdraw: not good");
869 ▾     for (int256 j = user.lastrewardsettled; j < currentday; j++)

```

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.

```

893     uint256 amount;
894     require(user.amount >= 0, "withdraw: not good");
895     require(now > user.lastdeposittime + rewardDuration, "Min Stake time is 24 hours");
896     amount = settleIncome(_pid, msg.sender);
897     amount = amount.add(user.unsettled);
898     require(amount > 0, "No Withdrawable amount");
899

```

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

```

914     UserInfo storage user = userInfo[_pid][msg.sender];
915
916     require(user.amount >= _amount, "withdraw: not good");
917     // require(
918     //     user.lastdeposittime + rewardDuration > now,
919     //     "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.

---

```
936 // Claim Reward will send token which earn us reward
937 function ClaimRewards(uint256 _pid) public {
938     // change method signature to accept pool name
939     UserInfo storage user = userInfo[_pid][msg.sender];
940     uint256 amount;
941
942     require(user.amount >= 0, "withdraw: not good");
943     require(now > user.lastdeposittime + rewardDuration, "reward not ready");
944
945     amount = settleIncome(_pid, msg.sender);
946     amount = amount.add(user.unsettled);
947
948     require(amount > 0, "No Withdrawable amount");
949 }
```

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

```
977 function safelyUseTokenTransfer(address _to, uint256 _amount) public {
978     uint256 ytheuseTokenBal = ytheuseToken.balanceOf(msg.sender);
979     require(ytheuseTokenBal >= _amount, "Low Balance ");
980     ytheuseToken.transfer(_to, _amount);
981 }
```

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

```
1030
1031 function dev(address _devaddr) public {
1032     require(msg.sender == devaddr, "dev: wut?");
1033     devaddr = _devaddr;
1034 }
```

---

## 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')

```
267 *   function isContract(address account) internal view returns (bool) {
268     // According to EIP-1052, 0x0 is the value returned for not-yet cre
269     // and 0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d
270     // for accounts without code, i.e. 'keccak256(')
271     bytes32 codehash;
272     bytes32 accountHash = 0xc5d2460186f7233c927e7db2dcc703c0e500b653ca8
273     // 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')

```
294 * https://solidity.readthedocs.io/en/v0.5.11/security-considerations.html
295 */
296 function sendValue(address payable recipient, uint256 amount)
297     require(address(this).balance >= amount, "Address: insufficient balance");
298
299     // solhint-disable-next-line avoid-low-level-calls, avoid-unsafe-low-level-calls
300     (bool success, ) = recipient.call{value: amount}("");
301     require(success, "Address: unable to send value, recipient may have deleted");
302 }
```

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')

```
361
362 function _functionCallWithValue(address target, bytes memory data, uint256 weiValue)
363     require(isContract(target), "Address: call to non-contract");
364     // solhint-disable-next-line avoid-low-level-calls
365     (bool success, bytes memory returndata) = target.call{value: weiValue}(data);
366     if (success) {
367         return returndata;
368     } else {
369         revert();
370     }
371 }
```

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')

```
388
389 function safeTransfer(
390     IERC20 token,
391     address to,
392     uint256 value
393 ) internal {
394     _callOptionalReturn(
395         token,
396         abi.encodeWithSelector(token.transfer.selector,
397             to, value));
398 }
```

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')

```
400     function safeTransferFrom(  
401         IERC20 token,  
402         address from,  
403         address to,  
404         uint256 value  
405     ) internal {  
406         _callOptionalReturn(  
407             token,  
408             abi.encodeWithSelector(token.transferFrom.selector,  
409             )
```

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')

```
419     function safeApprove(  
420         IERC20 token,  
421         address spender,  
422         uint256 value  
423     ) internal {  
424         // safeApprove should only be called when setting an initial allow  
425         // or when resetting it to zero. To increase and decrease it, use  
426         // 'safeIncreaseAllowance' and 'safeDecreaseAllowance'  
427         // solhint-disable-next-line max-line-length  
428         require(  
429             (spender == 0) || (token.allowance(spender, token) == 0))
```

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')

```
836     // should be for today only or expected daily token  
837     function pendingRewards(uint256 _pid, address _user)  
838         public  
839         view  
840         returns (uint256)  
841     {  
842         UserInfo storage user = userInfo[_pid][_user];
```

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')

```
976 |  
977 | function safeyTheuseTokenTransfer(address _to, uint256  
978 |     uint256 ytheuseTokenBal = ytheuseToken.balanceOf(a  
979 |     require(ytheuseTokenBal >= _amount, "Low Balance "  
980 |     ytheuseToken.transfer(_to, _amount);  
981 | }
```

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

### Function: - userData ('\_user')

```
982 |  
983 | function userData(uint256 _pid, address _user)  
984 |     public  
985 |     view  
986 |     returns (  
987 |         uint256 totalStakedTokens,  
988 |         uint256 stakedTokens
```

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

```
976
977 ▾ function safeyTheuseTokenTransfer(address _to, uint256 _amount) interr
978     uint256 ytheuseTokenBal = ytheuseToken.balanceOf(address(this));
979     require(ytheuseTokenBal >= _amount, "Low Balance ");
980     ytheuseToken.transfer(_to, _amount);
981 }
982
```

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

```
721     user.amount = user.amount.add(_amount);
722
723     user.lastdeposittime = now;
724     dailyData[_pid][currentday].totalInvestment += _amount;
725     user.lastrewardsettled = currentday;
726
727     emit Deposit(msg.sender, _pid, _amount);
```

### Function: - emitSetDailyData

```
741
742     function emitSetDailyData() public returns (bool emitted){
743         if (hasEmissionHappenedEver) {
744             if(now > lastGlobalEmissionDate + rewardDuration){
745                 setDailyData();
746                 return true;
747             }
748         }
```

### Function: - doSetDailyData

```
755
756     //only testing
757     function doSetDailyData() public onlyOwner returns (bool emitted){
758         if (hasEmissionHappenedEver) {
759             if(now > lastGlobalEmissionDate + rewardDuration){
760                 setDailyData();
761                 return true;
762             }
763         }
```

### Function: - setDailyData

```
813     //take out extra tokens if there were no staking on previous day
814     if (currentday != 0) {
815         if (dailyData[j][currentday - 1].totalInvestment == 0) {
816             extraTokens += dailyData[j][currentday - 1].totalrewards;
817             dailyData[j][currentday - 1].totalrewards = 0;
818         }
819     }
```

### Function: - pendingRewards

```
849     per = (user.amount).mul(FLOAT_SCALAR).div(dailyData[_pid][currentday].totalInvestment);
850
851     amount += (dailyData[_pid][currentday].totalrewards).mul(per).div(FLOAT_SCALAR);
852
853     return amount;
```

### Function: - settleIncome

```
867     uint256 per;
868     require(user.amount >= 0, "withdraw: not good");
869     for (int256 j = user.lastrewardsettled; j < currentday; j++) {
870         per = (user.amount).mul(FLOAT_SCALAR).div(dailyData[_pid][j].totalInvestment);
871         _amount += (dailyData[_pid][j].totalrewards).mul(per).div(FLOAT_SCALAR);
872     }
```

## Deployment on Testnet

POC (Proof of concept):

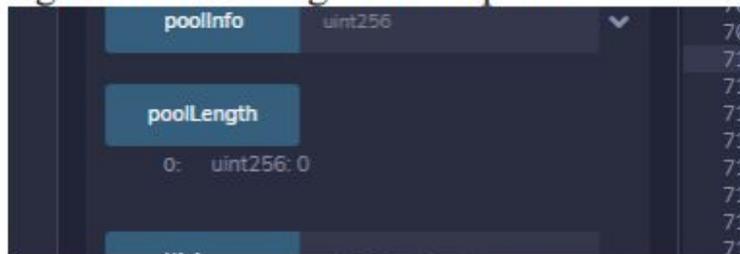
Function: - Deploy on testnet:-



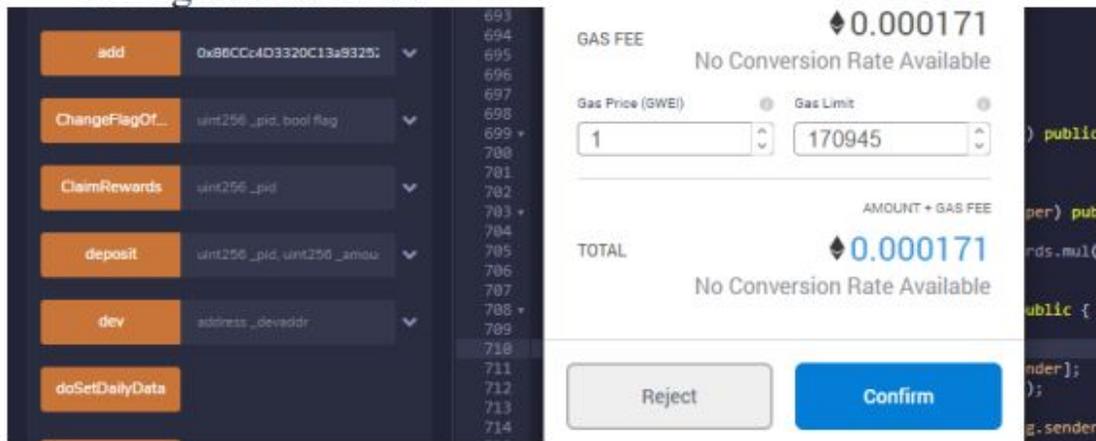
<https://goerli.etherscan.io/tx/0x42558c477862ada3f3cf618899e9ed073f2f68ec109d3da99c43421df3d076f3>

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

- PoolLength before adding token to pool.

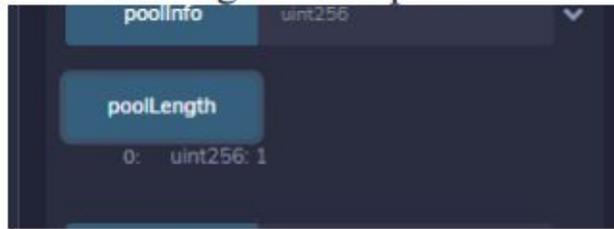


- Calling add function.

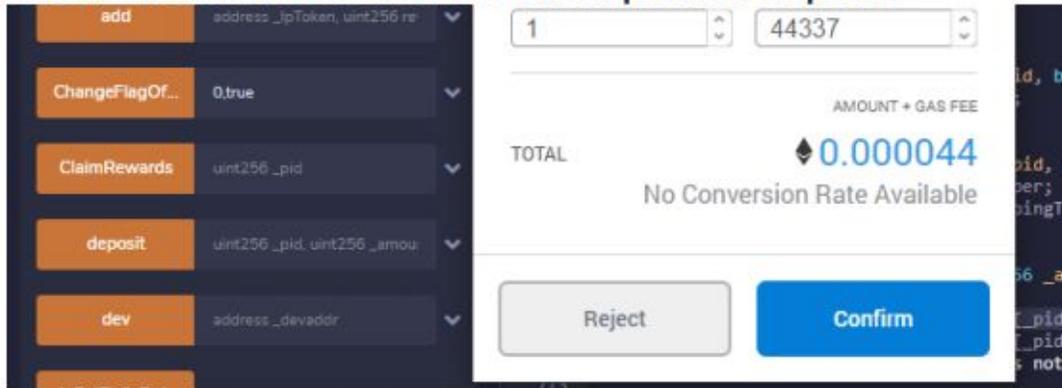


<https://goerli.etherscan.io/tx/0xc49974363a249c959ffa5dab233e7187ba356d703b09f9430f2548ec35a55700>

- PoolLength after adding token to pool.



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



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

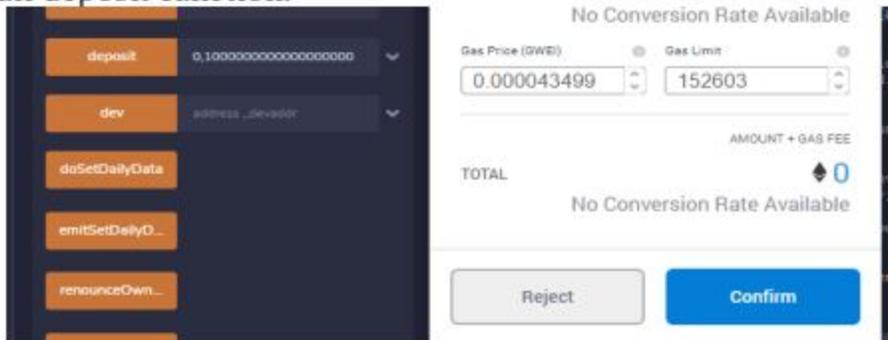
**Function: - deposit function call:-**

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



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

- Now call deposit function.



<https://goerli.etherscan.io/tx/0xefff6ed54a3b144c94451f9e2c8d668fcf220315f0c74d1fe70c660511d8e>





---

## 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**.

## Deployment

Update 31/10/2020: After reviewing every issue with the development team all code bugs/errors and improvement of code is correctly applied and fixed.

CTDSec provides a cybersecurity compliance certificate to yTheuse team.

Last file provided (**yTheuseMain\_Final\_v2.sol**) is safe to deploy and use.