CYBERSECURITY
J-COMPONENT
FINAL REPORT
ON

AUCTION WINNER DETERMINER WITH SOLIDITY AND REMIX IDE

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

NAME: KARTHIK K

REG.NO: 17BEC0695

SCHOOL: SENSE

FACULTY: PROF.LAVANYA K

SUBMISSION DATE:27/04/2020

ABSTRACT:

As the above title in front page suggest it is all about conducting an auction without any cause of ruckus and in a fair method for everyone and this can be done with the help of a blocks of code which constitutes for the whole process in a fair method, this will be elaborated in following report

The main reason for going with this method is that auction is cannot be corrupted as the bids can be seen by everyone and this is being more transparent throughout the process and also transactions are done then and there when everyone is there to witness the process, it is a genuine process where auction is transparent for everyone so even if the auction conductor cannot act biased for his own purpose of gain.

BRIEFING THE PROCESS

This process starts by having number distinct items that are for auction each have their own item id and own item token to identify them and at the same time we have address, remaining token and person Id for each bidders or person

Here the owner of smart contract will have his own details such as address id, etc. this will be distributed publicly for the transaction to happen

Now each or any number of persons can bid for any item but before that have to register for that particular item to bid for the item or to compete with other bidders, so for each bidders of the particular item that they are competing will be given the smart contract account details and the bidders for each will be given 5 tokens so each token will have unique values issued by the smart contract owner

For bidding we need two parameters those are item id and total count of the item(price of the item)

Here balance will be remaining token out of total count this balance will be updated to the bidders account and the token items detail will be sent to the person id

Now for revealing the winners we will access for the particular item with its item id after that there is a

calculation for determining the token id after determining the winning token id we map it with the address of the person it will show the address of the transaction of the winner id

PROGRAM CODE

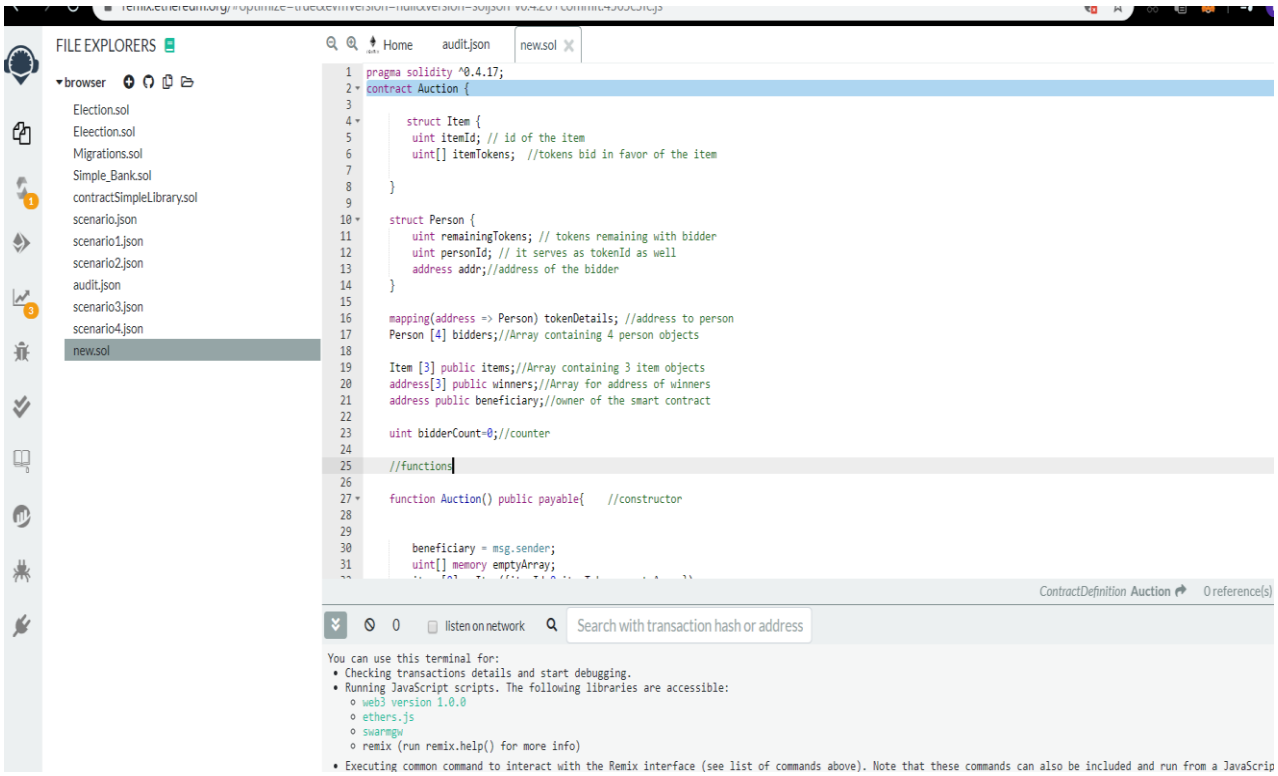
```
pragma solidity ^0.4.17;

contract Auction {


    struct Item {
     uint itemId; // id of the item
     uint[] itemTokens;  //tokens bid in favor of the item


  }


  struct Person {
     uint remainingTokens; // tokens remaining with bidder
     uint personId; // it serves as tokenId as well
     address addr;//address of the bidder
  }


  mapping(address => Person) tokenDetails; //address to person
  Person [4] bidders;//Array containing 4 person objects


  Item [3] public items;//Array containing 3 item objects
  address[3] public winners;//Array for address of winners
  address public beneficiary;//owner of the smart contract


  uint bidderCount=0;//counter


  //functions


  function Auction() public payable{    //constructor



     beneficiary = msg.sender;
```

```
        uint[] memory emptyArray;
        items[0] = Item({itemId:0,itemTokens:emptyArray});

        items[1] =  Item({itemId:1,itemTokens:emptyArray});
        items[2] =  Item({itemId:2,itemTokens:emptyArray});

    }


    function register() public payable{


        bidders[bidderCount].personId = bidderCount;


        bidders[bidderCount].addr = msg.sender;

        bidders[bidderCount].remainingTokens = 5; // only 5 tokens
        tokenDetails[msg.sender]=bidders[bidderCount];
        bidderCount++;
    }

    function bid(uint _itemId, uint _count) public payable {

        if (tokenDetails[msg.sender].remainingTokens < _count ||
bidders[bidderCount].remainingTokens == 0) revert();
        if (_itemId > 2) revert();

        uint balance = tokenDetails[msg.sender].remainingTokens - _count;

        tokenDetails[msg.sender].remainingTokens=balance;
```

```
        bidders[tokenDetails[msg.sender].personId].remainingTokens=balance;//updating the same
balance in bidders map.


        Item storage bidItem = items[_itemId];
        for(uint i=0; i<_count;i++) {
            bidItem.itemTokens.push(tokenDetails[msg.sender].personId);
        }
    }

    modifier onlyOwner {

        _;
    }



    function revealWinners() public onlyOwner{



        for (uint id = 0; id < 3; id++) {
            Item storage currentItem=items[id];
            if(currentItem.itemTokens.length != 0){
            // generate random# from block number
            uint randomIndex = (block.number / currentItem.itemTokens.length)%
currentItem.itemTokens.length;
            // Obtain the winning tokenId


            uint winnerId = currentItem.itemTokens[randomIndex];


            winners[id] = bidders[winnerId].addr;


            }
        } } }
```

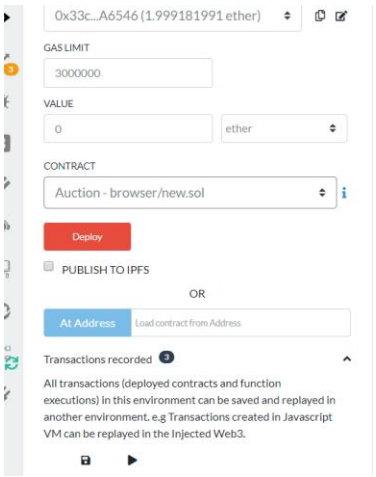
SCREENSHOTS

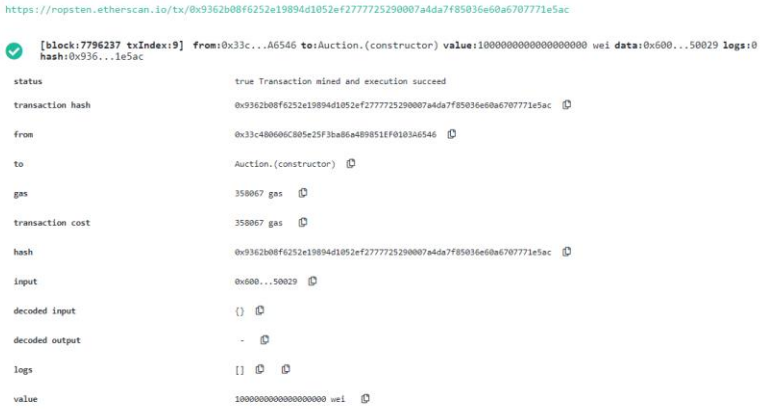
PROGRAM CODE



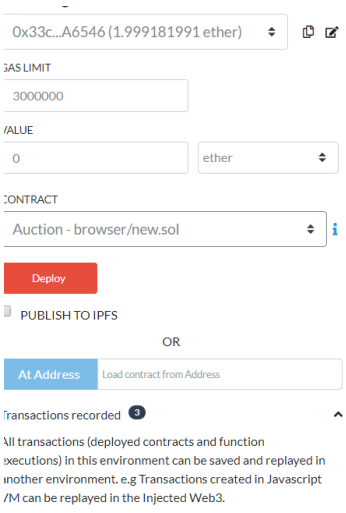
OUTPUT

RUNNING THE TRANSACTION

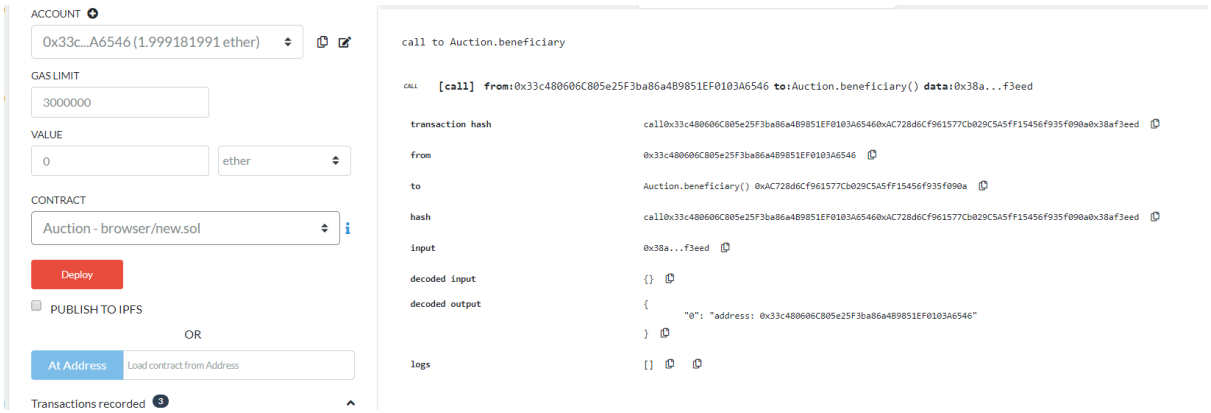




CALLING AUCTION REGISTER



CALLING AUCTION BENEFICIARY



CALLING AUCTION ITEMS

call to Auction.items

call **[call]** **from:**0x33c480606C805e25F3ba86a4B9851EF0103A6546 **to:**Auction.items(uint256) **data:**0xbfb...00000 Debug

transaction hash
call0x33c480606C805e25F3ba86a4B9851EF0103A65460xAC728d6Cf961577Cb029C5A5fF15456f935f090a0xbfb231d20000000000000000000000000000000000000000000000000000000000000000

from
0x33c480606C805e25F3ba86a4B9851EF0103A6546

to
Auction.items(uint256) 0xAC728d6Cf961577Cb029C5A5fF15456f935f090a

hash
call0x33c480606C805e25F3ba86a4B9851EF0103A65460xAC728d6Cf961577Cb029C5A5fF15456f935f090a0xbfb231d20000000000000000000000000000000000000000000000000000000000000000

input
0xbfb...00000

decoded input

```
{
        "uint256 ": {
                "_hex": "0x00"
        }
}
```

decoded output

```
{
        "0": "uint256: itemId 0"
}
```

logs
[]

AUCTION WINNERS REVEALING

CALLING THE AUCTION WINNERS

call to Auction.winners

CALL [call] from:0x33c480606C805e25F3ba86a4B9851EF0103A6546 to:Auction.winners(uint256) data:0xa2f...00000 Debug

transaction hash
call0x33c480606C805e25F3ba86a4B9851EF0103A65460x05211D291440e3D8022c24f8E8438B8038482944 0xa2fb11750000000000000000000000000000000000000000000000000000000000000000

from
0x33c480606C805e25F3ba86a4B9851EF0103A6546

to
Auction.winners(uint256) 0x05211D291440e3D8022c24f8E8438B8038482944

hash
call0x33c480606C805e25F3ba86a4B9851EF0103A65460x05211D291440e3D8022c24f8E8438B8038482944 0xa2fb11750000000000000000000000000000000000000000000000000000000000000000

input
0xa2f...00000

decoded input
```
{
        "uint256 ": {
                "_hex": "0x00"
        }
}
```

decoded output
```
{
        "0": "address: 0x0000000000000000000000000000000000000000"
}
```

logs
[]

CONCLUSION

As we can see the results from the above screenshots it is evident that this process is highly secured and transparent as corruption rate is very low in this model and this can be adapted anywhere anytime anyplace as this model has no restriction and the whole process takes place quite swiftly in a transparent manner this model is also implemented in many banking sectors and also certified by several governments around the world another takeaway from this model is that the transaction cost very low even for big amount of transaction that is why many business companies changing to this platform.

REFERENCES

https://www.youtube.com/watch?v=SSo_EIwHSd4

https://www.youtube.com/watch?v=ydp3sj0N0GQ

https://github.com/hillarymangena/Smart-Contract-for-a-Conventional-Auction/blob/master/Solidity%20source%20code