2CSDE93 - Blockchain Technology

Practical 9

Aim: Creating a Smart Contract

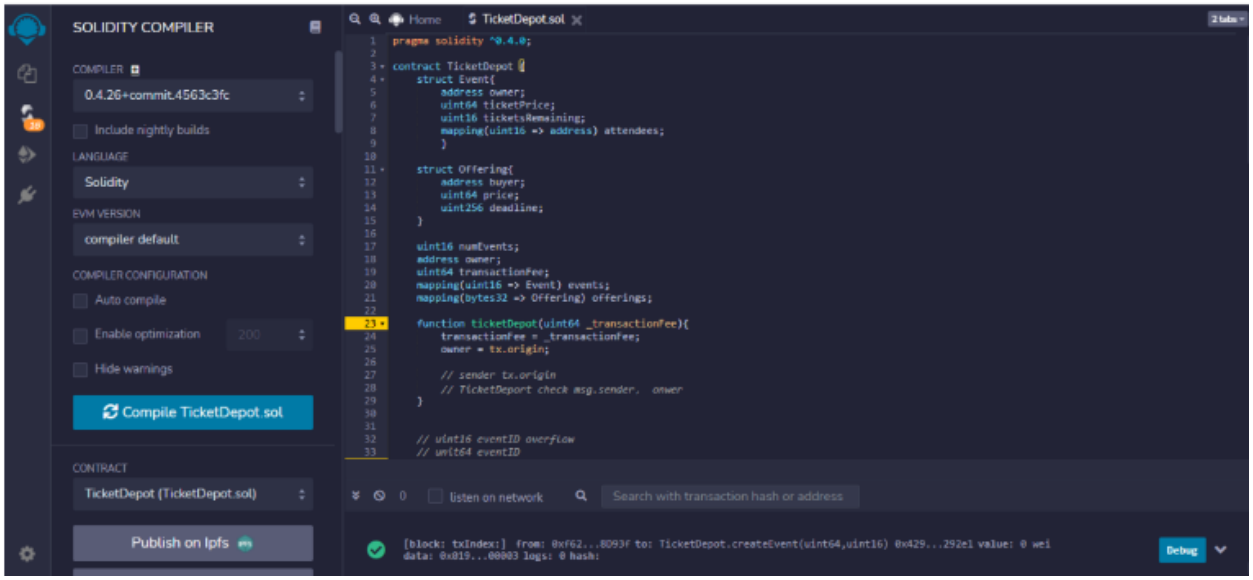
Author: Darshil Maru 20BCE514

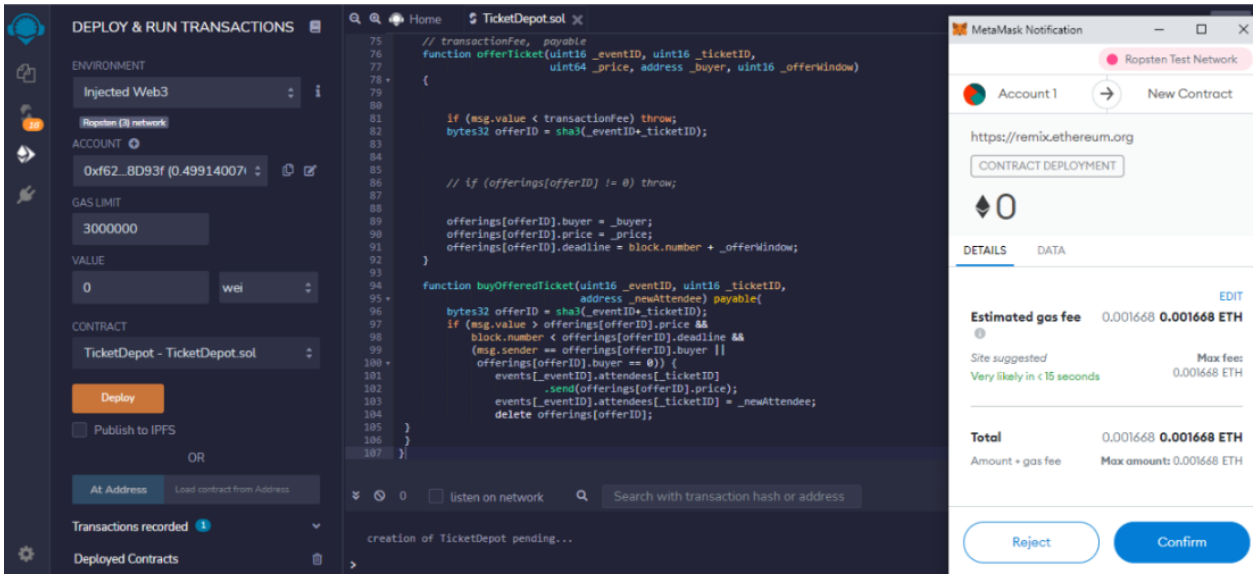
Aim: To write a Solidity contract that implements a distributed ticket sales system. Anybody can create an event (specifying the initial price and number of tickets). Anybody can then purchase one of the initial tickets or sell those tickets peer-topeer. At the event, gate agents will check that each attendee is listed in the final attendees list on the blockchain. (Ethereum programming).

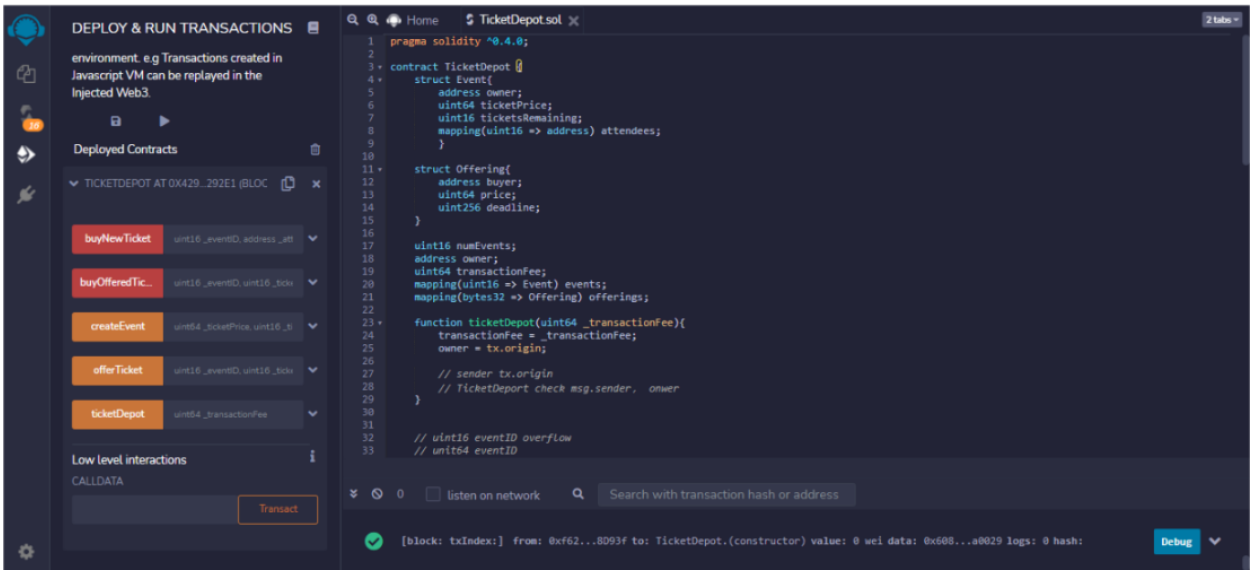
```
pragma solidity ^0.4.0;
contract TicketDepot {

    struct Event{
        address owner;
        uint64 ticketPrice;
        uint16 ticketsRemaining;
        mapping(uint16 => address) attendees;
    }

    struct Offering{
        address buyer;
        uint64 price;
        uint256 deadline;
    }

    uint16 numEvents;
    address owner;
    uint64 transactionFee;
    mapping(uint16 => Event) events;
    mapping(bytes32 => Offering) offerings;
    function ticketDepot(uint64 _transactionFee){
        transactionFee = _transactionFee;
        owner = tx.origin;
    }

    function createEvent(uint64 _ticketPrice, uint16 _ticketsAvailable)
returns (uint16 eventID){
        numEvents++;
        events[numEvents].owner = tx.origin;
        events[numEvents].ticketPrice = _ticketPrice;
        events[numEvents].ticketsRemaining = _ticketsAvailable;
        return numEvents;
```

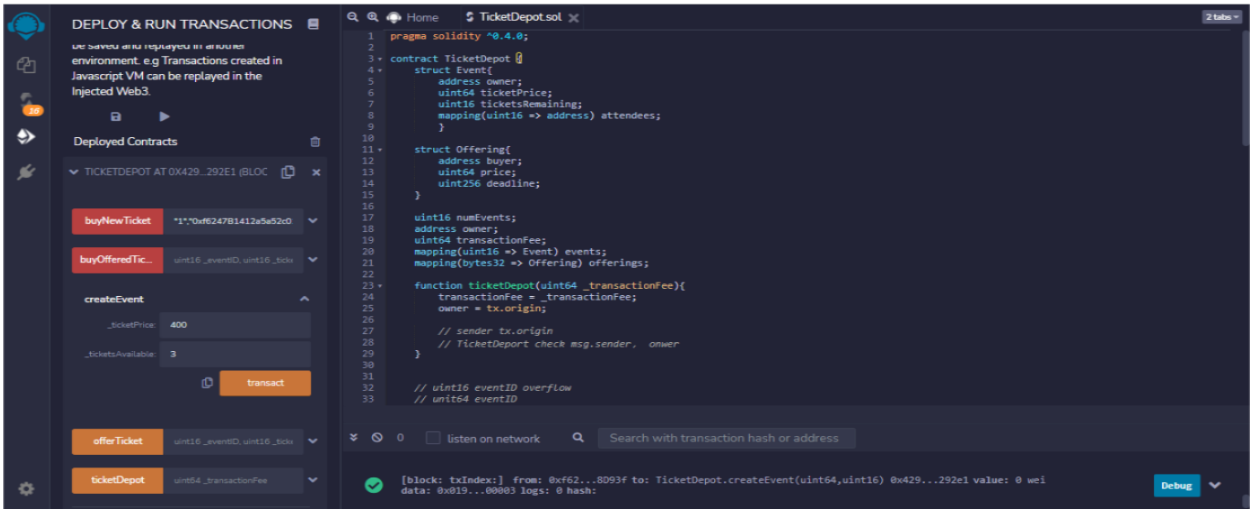
```
    }
    modifier ticketsAvailable(uint16 _eventID){
        _;
        if (events[_eventID].ticketsRemaining <= 0) throw;
    }
    function buyNewTicket(uint16 _eventID, address _attendee)
ticketsAvailable(_eventID) payable returns (uint16 ticketID){
        if (msg.sender == events[_eventID].owner || msg.value >
events[_eventID].ticketPrice + transactionFee){
            ticketID = events[_eventID].ticketsRemaining--;
            events[_eventID].attendees[ticketID] = _attendee;
            events[_eventID].owner.send(msg.value - transactionFee);
            return ticketID;
        }
    }
    function offerTicket(uint16 _eventID, uint16 _ticketID, uint64 _price,
address _buyer, uint16 _offerWindow) {
        if (msg.value < transactionFee) throw;
        bytes32 offerID = sha3(_eventID+_ticketID);
        //if (offerings[offerID] != 0) throw;
        offerings[offerID].buyer = _buyer;
        offerings[offerID].price = _price;
        offerings[offerID].deadline = block.number + _offerWindow;
    }

    function buyOfferedTicket(uint16 _eventID, uint16 _ticketID, address
_newAttendee) payable{
        bytes32 offerID = sha3(_eventID+_ticketID);
        if (msg.value > offerings[offerID].price &&
            block.number < offerings[offerID].deadline &&
            (msg.sender == offerings[offerID].buyer ||
            offerings[offerID].buyer == 0)) {

            events[_eventID].attendees[_ticketID]
                .send(offerings[offerID].price);
                events[_eventID].attendees[_ticketID] = _newAttendee;
                delete offerings[offerID];
        }
    }
}
```

Output:


SOLIDITY COMPILER
COMPILER
0.4.26+commit.4563c3fc
Include nightly builds
LANGUAGE
Solidity
EVM VERSION
compiler default
COMPILER CONFIGURATION
Auto compile
Enable optimization
Hide warnings
Compile TicketDepot.sol
CONTRACT
TicketDepot (TicketDepot.sol)
Publish on Ipfs
Home
TicketDepot.sol
pragma solidity ^0.4.0;
contract TicketDepot {
struct Event{
address owner;
uint64 ticketPrice;
uint16 ticketsRemaining;
mapping(uint16 => address) attendees;
}
struct Offering{
address buyer;
uint64 price;
uint256 deadline;
}
uint16 numEvents;
address owner;
uint64 transactionFee;
mapping(uint16 => Event) events;
mapping(bytes32 => Offering) offerings;
function ticketDepot(uint64 _transactionFee){
transactionFee = _transactionFee;
owner = tx.origin;
// sender tx.origin
// TicketDeport check msg.sender, owner
}
// uint16 eventID overflow
// uint64 eventID
listen on network
Search with transaction hash or address
Debug


DEPLOY & RUN TRANSACTIONS
ENVIRONMENT
Injected Web3
Ropsten (3) network
ACCOUNT
0xf62...8D93f (0.49914007
GAS LIMIT
3000000
VALUE
0
wei
CONTRACT
TicketDepot - TicketDepot.sol
Deploy
Publish to IPFS
OR
At Address
Load contract from Address
Transactions recorded
Deployed Contracts
Home
TicketDepot.sol
// transactionFee, payable
function offerTicket(uint16 _eventID, uint16 _ticketID,
uint64 _price, address _buyer, uint16 _offerWindow)
{
if (msg.value < transactionFee) throw;
bytes32 offerID = sha3(_eventID+_ticketID);
// if (offerings[offerID] != 0) throw;
offerings[offerID].buyer = _buyer;
offerings[offerID].price = _price;
offerings[offerID].deadline = block.number + _offerWindow;
}
function buyOfferedTicket(uint16 _eventID, uint16 _ticketID,
address _newAttendee) payable{
bytes32 offerID = sha3(_eventID+_ticketID);
if (msg.value > offerings[offerID].price &&
block.number < offerings[offerID].deadline &&
(msg.sender == offerings[offerID].buyer ||
offerings[offerID].buyer == 0)) {
events[_eventID].attendees[_ticketID]
.send(offerings[offerID].price);
events[_eventID].attendees[_ticketID] = _newAttendee;
delete offerings[offerID];
}
}
}
listen on network
Search with transaction hash or address
creation of TicketDepot pending...
MetaMask Notification
Ropsten Test Network
Account 1
New Contract
https://remix.ethereum.org
CONTRACT DEPLOYMENT
0
DETAILS
DATA
EDIT
Estimated gas fee
0.001668 0.001668 ETH
Site suggested
Very likely in < 15 seconds
Max fee:
0.001668 ETH
Total
0.001668 0.001668 ETH
Amount + gas fee
Max amount: 0.001668 ETH
Reject
Confirm


DEPLOY & RUN TRANSACTIONS
environment. e.g Transactions created in Javascript VM can be replayed in the Injected Web3.
Deployed Contracts
TICKETDEPOT AT 0X429...292E1 (BLOC
buyNewTicket
buyOfferedTic...
createEvent
offerTicket
ticketDepot
Low level interactions
CALLDATA
Transact
Home
TicketDepot.sol
2 tabs
pragma solidity ^0.4.0;
contract TicketDepot {
struct Event{
address owner;
uint64 ticketPrice;
uint16 ticketsRemaining;
mapping(uint16 => address) attendees;
}
struct Offering{
address buyer;
uint64 price;
uint256 deadline;
}
uint16 numEvents;
address owner;
uint64 transactionFee;
mapping(uint16 => Event) events;
mapping(bytes32 => Offering) offerings;
function ticketDepot(uint64 _transactionFee){
transactionFee = _transactionFee;
owner = tx.origin;
// sender tx.origin
// TicketDeport check msg.sender, onwer
}
// uint16 eventID overflow
// uint64 eventID
listen on network
Search with transaction hash or address
[block: txIndex:] from: 0xf62...8D93f to: TicketDepot.(constructor) value: 0 wei data: 0x608...a0029 logs: 0 hash:
Debug


DEPLOY & RUN TRANSACTIONS
environment. e.g Transactions created in Javascript VM can be replayed in the Injected Web3.
Deployed Contracts
TICKETDEPOT AT 0X429...292E1 (BLOC
buyNewTicket
"1","0xf6247B1412a5a52c0
buyOfferedTic...
createEvent
_ticketPrice:
400
_ticketsAvailable:
3
transact
offerTicket
ticketDepot
Home
TicketDepot.sol
2 tabs
pragma solidity ^0.4.0;
contract TicketDepot {
struct Event{
address owner;
uint64 ticketPrice;
uint16 ticketsRemaining;
mapping(uint16 => address) attendees;
}
struct Offering{
address buyer;
uint64 price;
uint256 deadline;
}
uint16 numEvents;
address owner;
uint64 transactionFee;
mapping(uint16 => Event) events;
mapping(bytes32 => Offering) offerings;
function ticketDepot(uint64 _transactionFee){
transactionFee = _transactionFee;
owner = tx.origin;
// sender tx.origin
// TicketDeport check msg.sender, onwer
}
// uint16 eventID overflow
// uint64 eventID
listen on network
Search with transaction hash or address
[block: txIndex:] from: 0xf62...8D93f to: TicketDepot.createEvent(uint64,uint16) 0x429...292e1 value: 0 wei
data: 0x019...00003 logs: 0 hash:
Debug


DEPLOY & RUN TRANSACTIONS
buyNewTicket
"1","0xf6247B1412a5a52c0
buyOfferedTic...
createEvent
_ticketPrice:
"400"
_ticketsAvailable:
"3"
transact
offerTicket
_eventID:
uint16
_ticketID:
uint16
_price:
uint64
_buyer:
address
_offerWindow:
uint16
transact
Home
TicketDepot.sol
2 tabs
pragma solidity ^0.4.0;
contract TicketDepot {
struct Event{
address owner;
uint64 ticketPrice;
uint16 ticketsRemaining;
mapping(uint16 => address) attendees;
}
struct Offering{
address buyer;
uint64 price;
uint256 deadline;
}
uint16 numEvents;
address owner;
uint64 transactionFee;
mapping(uint16 => Event) events;
mapping(bytes32 => Offering) offerings;
function ticketDepot(uint64 _transactionFee){
transactionFee = _transactionFee;
owner = tx.origin;
// sender tx.origin
// TicketDeport check msg.sender, onwer
}
// uint16 eventID overflow
// uint64 eventID
listen on network
Search with transaction hash or address
[block: txIndex:] from: 0xf62...8D93f to: TicketDepot.createEvent(uint64,uint16) 0x429...292e1 value: 0 wei
data: 0x019...00003 logs: 0 hash:
Debug