Lottereum

Contents

1 Introduction

A contract for a decentralized fair lottery was constructed to fulfill the requirements that ensure winner selection is not biased by any single party:

- each participant contributes a secret input along with their bet in phase 1
- no further input can be made after secret inputs are revealed in phase 2
- secret inputs are combined to select a random winner
- winners are selected according to their bet ratio
- a honesty deposit is required per buy-in
- the pot for the honesty deposit is distributed to participants who reveal their secret inputs in phase 2
- the honesty deposit increases in the case of dishonest play, and decreases in the case of honest play
- each phase should have some way to track gameplay length
- anybody can call for the reward function to select a winner and complete the payout in phase 3
- anybody should be able to create concurrent lottery rounds

2 Methodology

2.1 Phase 1 - Secret Input and Betting

Each buy-in is stored as a `struct Ticket` that is mapped to the player's address within each `struct Game`. The user's 32-byte input is hashed and stored in the ticket.

2.2 Phase 2 - Resolution of Secret Inputs

When the time allocated for phase 1 is over, `resolveGame()` is allowed to be called by players. Players will have the same amount of time as allocated to phase 1 to submit their initial secret inputs. The hashes of the submissions are compared to their ticket to determine their honesty. Honest players are then added to the honestPlayers list.

User inputs of honest players are combined by `mergeHash()`. This interleaves `bytes32` positions of `game.combinedInput`, which is a running hash of inputs, with the `bytes32` submitted input, resulting in a length of 64 bytes. The combined input is re-hashed using `keccak256` again resulting in a 32-byte long hash.

2.3 Phase 3 - Deposit Disbursement and Winner Selection

When the time allocated for phase 2 is over, `getPayout()` is allowed to be called by players. If no winner has been selected yet, `depositPot` is disbursed equally to the list of `honestPlayers`, and `currentDeposit` is adjusted for new rounds.

`game.combinedInput` is casted to `uint256` and the modulus against `game.jackpot + 1` is used to obtain a `combinedRandom` number that falls within the range of `game.jackpot`. The running total of bets made by each honest player is then summed up until it exceeds the `combinedRandom` number, which determines the lottery winner.

2.4 Concurrent Lottery Rounds

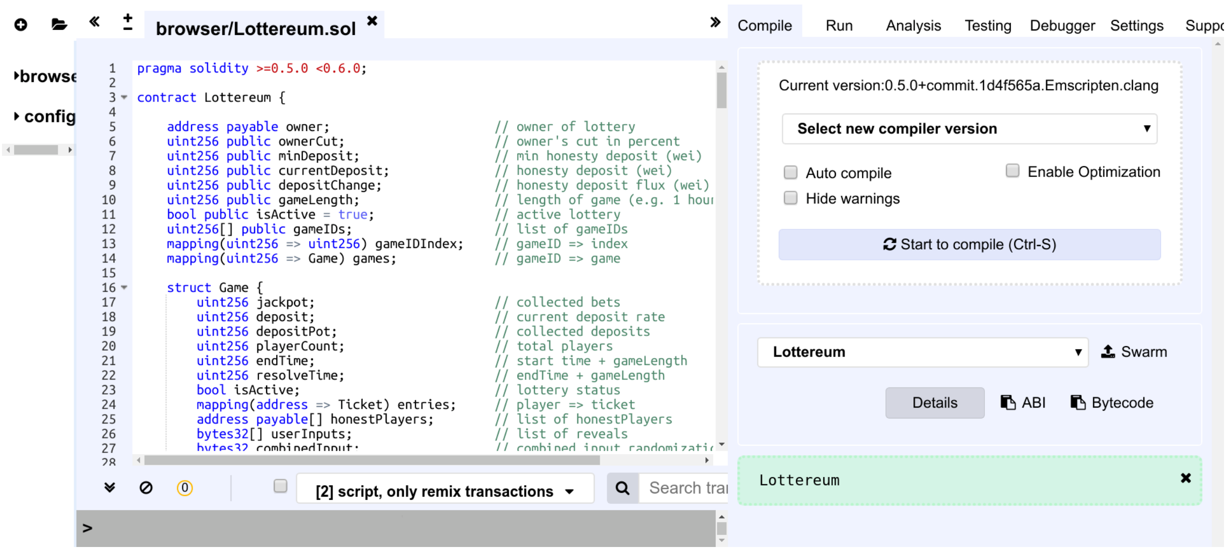
Games are identified by their current block's timestamp in epoch time (`now`). Multiple active `gameIDs` are tracked using an array. The array of games is trimmed whenever a game expires.

2.5 Tracking Gameplay Length

Each phase of a game is tracked by adding time in seconds to the `gameID`. The game's `endTime` and `resolveTime` compared to `now` determines when a function call for the respective phase is allowed to execute.

3 Results

3.1 Successful Compilation



```
pragma solidity >=0.5.0 <0.6.0;

contract Lottereum {

    address payable owner;                      // owner of lottery
    uint256 public ownerCut;                    // owner's cut in percent
    uint256 public minDeposit;                  // min honesty deposit (wei)
    uint256 public currentDeposit;              // honesty deposit (wei)
    uint256 public depositChange;               // honesty deposit flux (wei)
    uint256 public gameLength;                  // length of game (e.g. 1 hour
    bool public isActive = true;                // active lottery
    uint256[] public gameIDs;                   // list of gameIDs
    mapping(uint256 => uint256) gameIDIndex;    // gameID => index
    mapping(uint256 => Game) games;             // gameID => game

    struct Game {
        uint256 jackpot;                        // collected bets
        uint256 deposit;                        // current deposit rate
        uint256 depositPot;                     // collected deposits
        uint256 playerCount;                    // total players
        uint256 endTime;                        // start time + gameLength
        uint256 resolveTime;                    // endTime + gameLength
        bool isActive;                          // lottery status
        mapping(address => Ticket) entries;     // player => ticket
        address payable[] honestPlayers;        // list of honestPlayers
        bytes32[] userInputs;                   // list of reveals
        bytes32 combinedInput;                  // combined input randomizati
```

Current version:0.5.0+commit.1d4f565a.Emscripten.clang

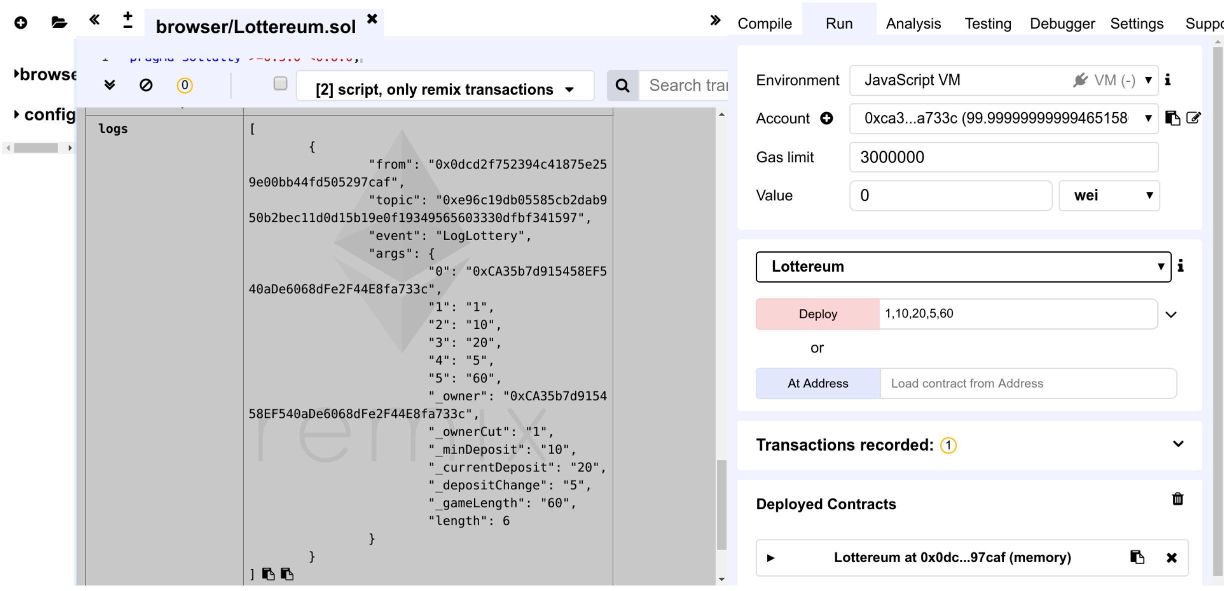
Lottereum

Lottereum

Compiled with solidity 0.5.0

`solidity >= 0.5.0` was chosen to adapt to breaking changes and to keep the contract code future-proof for as long as possible.

3.2 Deploying the Contract



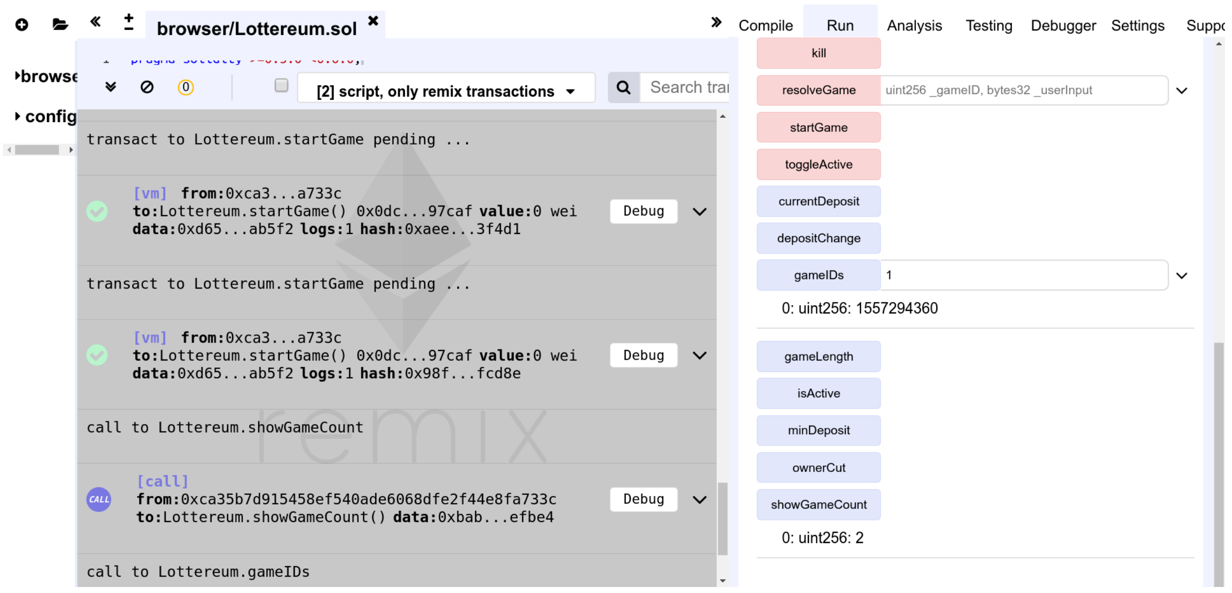
```
logs    [
                {
                        "from": "0x0dcd2f752394c41875e25
        9e00bb44fd505297caf",
                        "topic": "0xe96c19db05585cb2dab9
        50b2bec11d0d15b19e0f19349565603330dfbf341597",
                        "event": "LogLottery",
                        "args": {
                                "0": "0xCA35b7d915458EF5
        40aDe6068dFe2F44E8fa733c",
                                "1": "1",
                                "2": "10",
                                "3": "20",
                                "4": "5",
                                "5": "60",
                                "_owner": "0xCA35b7d9154
        58EF540aDe6068dFe2F44E8fa733c",
                                "_ownerCut": "1",
                                "_minDeposit": "10",
                                "_currentDeposit": "20",
                                "_depositChange": "5",
                                "_gameLength": "60",
                                "length": 6
                        }
                }
        ]
```

Environment JavaScript VM VM (-)
Account 0xca3...a733c (99.99999999999465158
Gas limit 3000000
Value 0 wei
Lottereum
Deploy 1,10,20,5,60
Lottereum at 0x0dc...97caf (memory)

Contract deployment

The contract can be deployed with user-defined variables. In the above figure, the contract was deployed with the owner taking 1% of the winnings, a minimum honesty deposit of 10 wei, a current deposit of 20 wei, a fluctuation of 5 wei, and a game time of 60 seconds.

3.3 Starting Multiple Games


```
transact to Lottereum.startGame pending ...

[vm] from:0xca3...a733c
to:Lottereum.startGame() 0x0dc...97caf value:0 wei
data:0xd65...ab5f2 logs:1 hash:0xaee...3f4d1

transact to Lottereum.startGame pending ...

[vm] from:0xca3...a733c
to:Lottereum.startGame() 0x0dc...97caf value:0 wei
data:0xd65...ab5f2 logs:1 hash:0x98f...fcd8e

call to Lottereum.showGameCount

[call]
from:0xca35b7d915458ef540ade6068dfe2f44e8fa733c
to:Lottereum.showGameCount() data:0xbab...efbe4

call to Lottereum.gameIDs
```

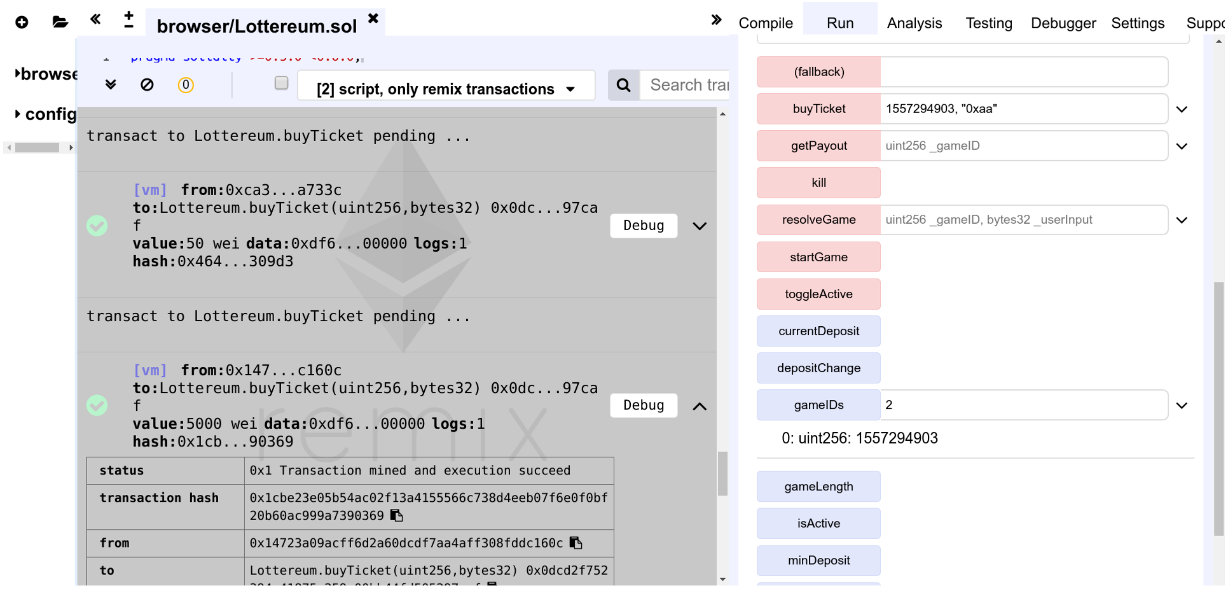
```
gameIDs 1
0: uint256: 1557294360

showGameCount
0: uint256: 2
```

Multiple games can be started

Concurrent games can be started by anybody. The games are tracked by their `GameIDs`, which can be retried by checking their index limited by `showGameCount`.

3.4 Buying Tickets


```
transact to Lottereum.buyTicket pending ...

[vm] from:0xca3...a733c
to:Lottereum.buyTicket(uint256,bytes32) 0x0dc...97ca
f
value:50 wei data:0xdf6...00000 logs:1
hash:0x464...309d3

transact to Lottereum.buyTicket pending ...

[vm] from:0x147...c160c
to:Lottereum.buyTicket(uint256,bytes32) 0x0dc...97ca
f
value:5000 wei data:0xdf6...00000 logs:1
hash:0x1cb...90369
```

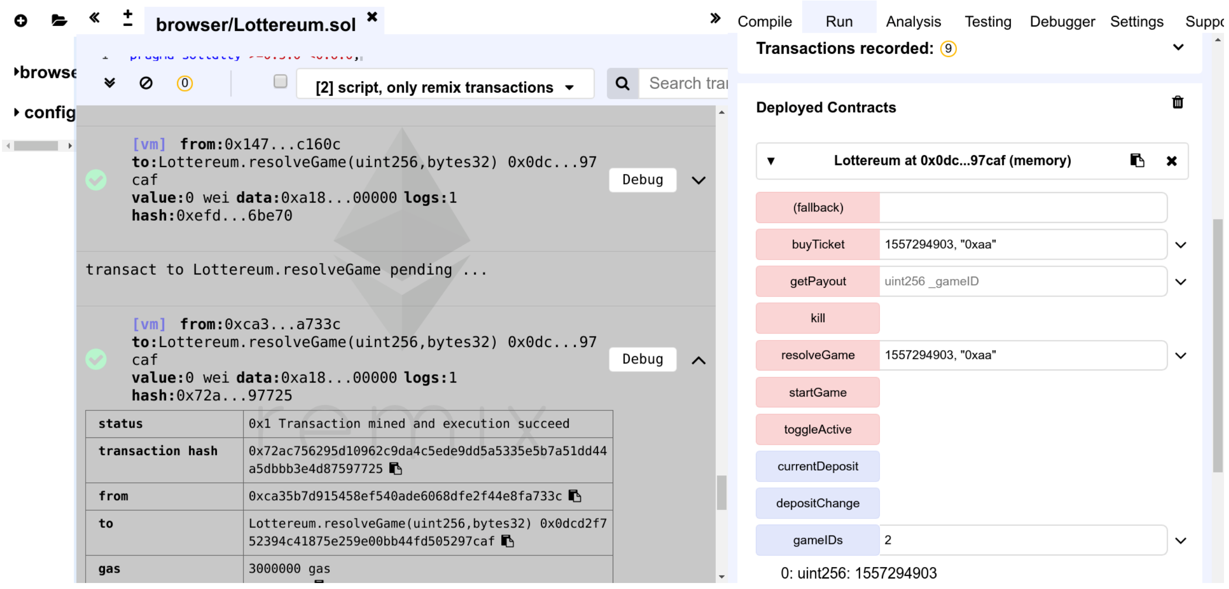
status	0x1 Transaction mined and execution succeed
transaction hash	0x1cbe23e05b54ac02f13a4155566c738d4eeb07f6e0f0bf20b60ac999a7390369
from	0x14723a09acff6d2a60dcdf7aa4aff308fddc160c
to	Lottereum.buyTicket(uint256,bytes32) 0x0dcd2f752

```
buyTicket 1557294903, "0xaa"
gameIDs 2
0: uint256: 1557294903
```

Buying ticket from different accounts

Two tickets were purchased by two accounts. `0xca3...a733c` purchased a ticket worth 50 wei, while `0x147...c160c` purchased a ticket worth 5000 wei. 20 wei will be taken from each player for their deposit. Each player also has to provide a `bytes32` input during their purchase, which is hashed and stored in their `Ticket`.

3.5 Resolving Secret Inputs



```
[vm] from:0x147...c160c
to:Lottereum.resolveGame(uint256,bytes32) 0x0dc...97
caf
value:0 wei data:0xa18...00000 logs:1
hash:0xefd...6be70

transact to Lottereum.resolveGame pending ...

[vm] from:0xca3...a733c
to:Lottereum.resolveGame(uint256,bytes32) 0x0dc...97
caf
value:0 wei data:0xa18...00000 logs:1
hash:0x72a...97725
```

status	0x1 Transaction mined and execution succeed
transaction hash	0x72ac756295d10962c9da4c5ede9dd5a5335e5b7a51dd44a5dbbb3e4d87597725
from	0xca35b7d915458ef540ade6068dfe2f44e8fa733c
to	Lottereum.resolveGame(uint256,bytes32) 0x0dcd2f752394c41875e259e00bb44fd505297caf
gas	3000000 gas

Transactions recorded: 9

Deployed Contracts

Lottereum at 0x0dc...97caf (memory)

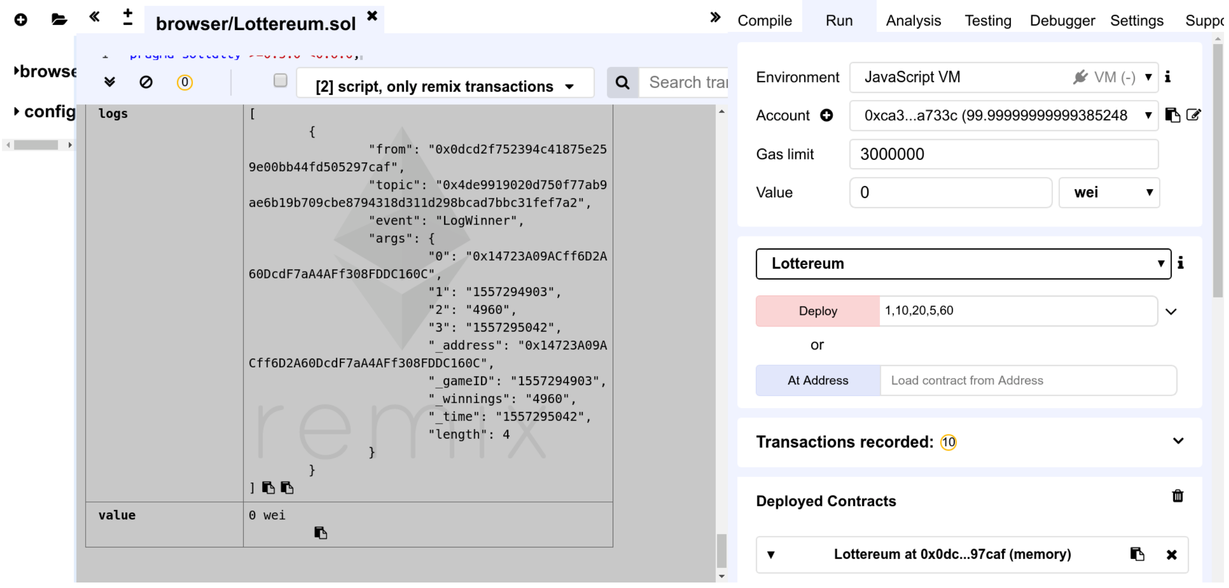
(fallback)	
buyTicket	1557294903, "0xaa"
getPayout	uint256 _gameID
kill	
resolveGame	1557294903, "0xaa"
startGame	
toggleActive	
currentDeposit	
depositChange	
gameIDs	2

0: uint256: 1557294903

Resolving inputs from different accounts

Both accounts that purchased a ticket send in their inputs again during the game resolution phase in order to participate in the draw.

3.6 Getting Payouts



```
logs    [
                {
                        "from": "0x0dcd2f752394c41875e25
        9e00bb44fd505297caf",
                        "topic": "0x4de9919020d750f77ab9
        ae6b19b709cbe8794318d311d298bcad7bbc31fef7a2",
                        "event": "LogWinner",
                        "args": {
                                "0": "0x14723A09ACff6D2A
        60DcdF7aA4AFf308FDDC160C",
                                "1": "1557294903",
                                "2": "4960",
                                "3": "1557295042",
                                "_address": "0x14723A09A
        Cff6D2A60DcdF7aA4AFf308FDDC160C",
                                "_gameID": "1557294903",
                                "_winnings": "4960",
                                "_time": "1557295042",
                                "length": 4
                        }
                }
        ]
value   0 wei
```

Environment	JavaScript VM
Account	0xca3...a733c (99.99999999999385248
Gas limit	3000000
Value	0 wei

Lottereum

Deploy 1,10,20,5,60

or

At Address Load contract from Address

Transactions recorded: 10

Deployed Contracts

Lottereum at 0x0dc...97caf (memory)

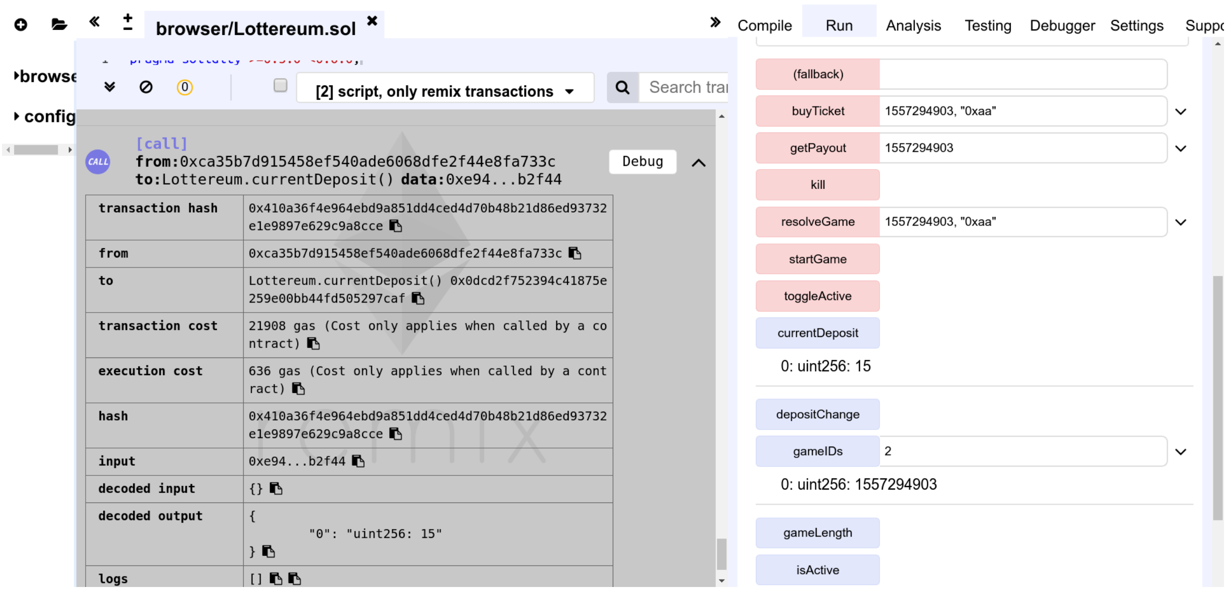
Requesting for the winner and getting payouts

The screenshot shows the first account `0xca3...a733c` requesting for payout during the payout phase. Unsurprisingly, `0x147...c160c` who bet a comparatively large amount was chosen as the winner, with a payout of 4960 wei. This amount is a result of the following:

Players	Bets	Deposit	Cut	Total
`0xca3`	50	20	0.3	30
`0x147`	5000	20	49.8	4930

The game cleans itself up at the end of `getPayout()`, removing itself from the array of `GameIDs` in order to get some gas refund.

3.7 Honesty Deposit



```
[call]
from:0xca35b7d915458ef540ade6068dfe2f44e8fa733c
to:Lottereum.currentDeposit() data:0xe94...b2f44
```

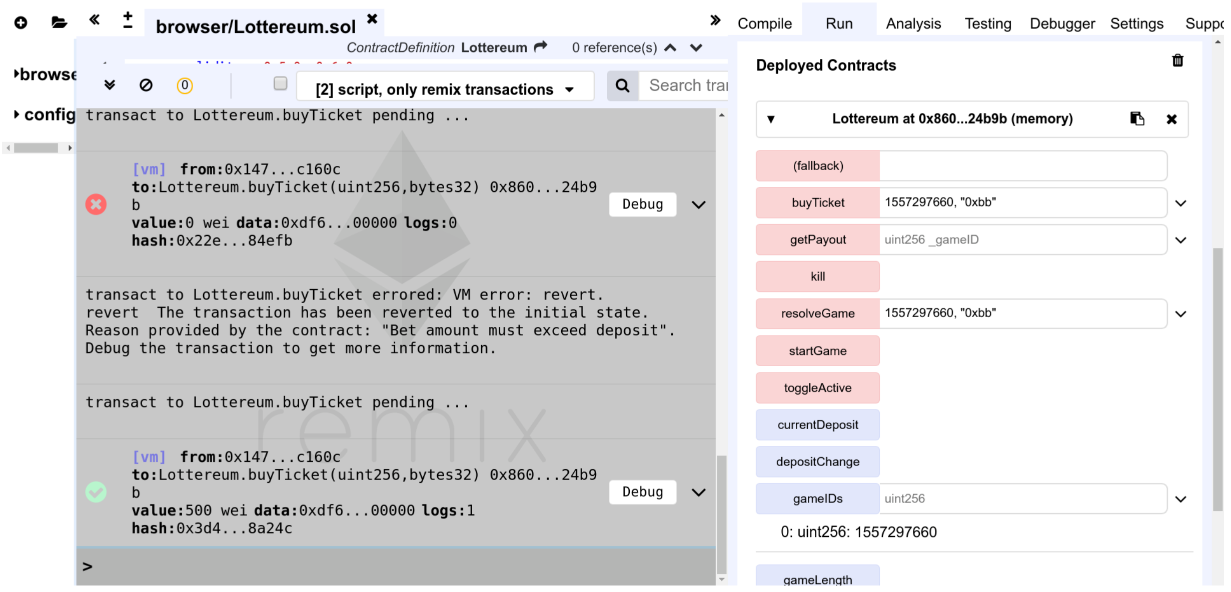
transaction hash	0x410a36f4e964ebd9a851dd4ced4d70b48b21d86ed93732e1e9897e629c9a8cce
from	0xca35b7d915458ef540ade6068dfe2f44e8fa733c
to	Lottereum.currentDeposit() 0x0dcd2f752394c41875e259e00bb44fd505297caf
transaction cost	21908 gas (Cost only applies when called by a contract)
execution cost	636 gas (Cost only applies when called by a contract)
hash	0x410a36f4e964ebd9a851dd4ced4d70b48b21d86ed93732e1e9897e629c9a8cce
input	0xe94...b2f44
decoded input	{}
decoded output	{ "0": "uint256: 15" }
logs	[]

(fallback)	
buyTicket	1557294903, "0xaa"
getPayout	1557294903
kill	
resolveGame	1557294903, "0xaa"
startGame	
toggleActive	
currentDeposit	
	0: uint256: 15
depositChange	
gameIDs	2
	0: uint256: 1557294903
gameLength	
isActive	

Reduced honesty deposit

The honesty deposit reduces to 15 wei as players were honest.

3.8 Dishonest Play



```
transact to Lottereum.buyTicket pending ...

[vm] from:0x147...c160c
to:Lottereum.buyTicket(uint256,bytes32) 0x860...24b9
b
value:0 wei data:0xdf6...00000 logs:0
hash:0x22e...84efb

transact to Lottereum.buyTicket errored: VM error: revert.
revert  The transaction has been reverted to the initial state.
Reason provided by the contract: "Bet amount must exceed deposit".
Debug the transaction to get more information.

transact to Lottereum.buyTicket pending ...

[vm] from:0x147...c160c
to:Lottereum.buyTicket(uint256,bytes32) 0x860...24b9
b
value:500 wei data:0xdf6...00000 logs:1
hash:0x3d4...8a24c
>
```

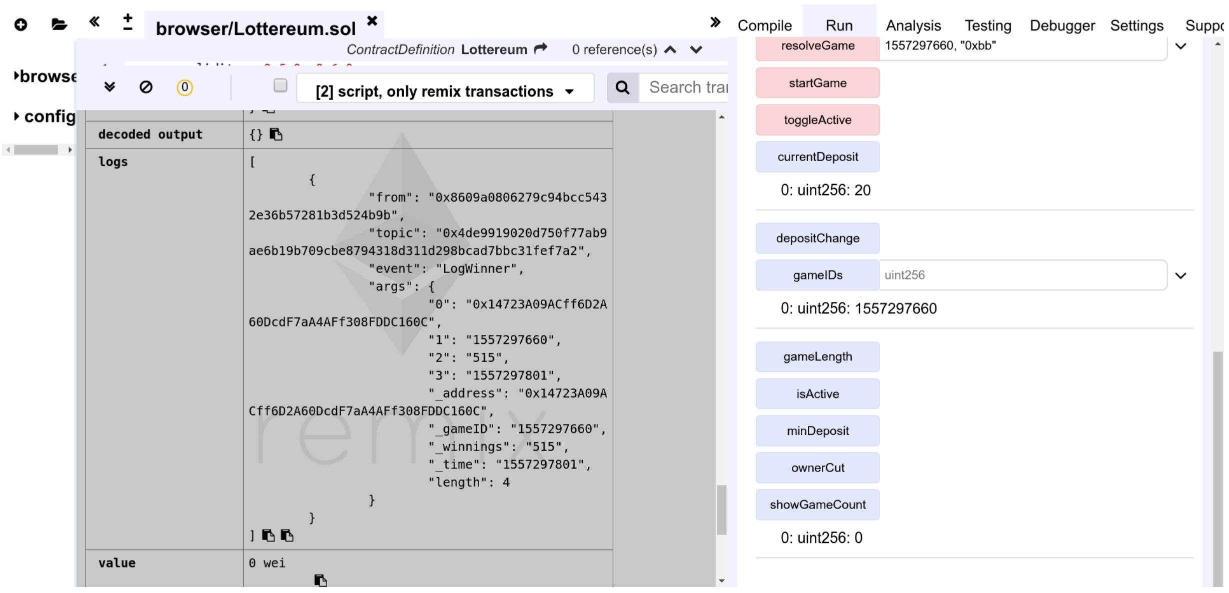
Deployed Contracts

Lottereum at 0x860...24b9b (memory)

(fallback)	
buyTicket	1557297660, "0xbb"
getPayout	uint256 _gameID
kill	
resolveGame	1557297660, "0xbb"
startGame	
toggleActive	
currentDeposit	
depositChange	
gameIDs	uint256
	0: uint256: 1557297660
gameLength	

Second round of buy-ins

A second round of lottery was conducted to demonstrate dishonest play. Note the buy-in does not occur if the bet amount falls below the deposit (currently at 15 wei).



```
decoded output   {}
logs             [
                         {
                                 "from": "0x8609a0806279c94bcc543
                 2e36b57281b3d524b9b",
                                 "topic": "0x4de9919020d750f77ab9
                 ae6b19b709cbe8794318d311d298bcad7bbc31fef7a2",
                                 "event": "LogWinner",
                                 "args": {
                                         "0": "0x14723A09ACff6D2A
                 60DcdF7aA4AFf308FDDC160C",
                                         "1": "1557297660",
                                         "2": "515",
                                         "3": "1557297801",
                                         "_address": "0x14723A09A
                 Cff6D2A60DcdF7aA4AFf308FDDC160C",
                                         "_gameID": "1557297660",
                                         "_winnings": "515",
                                         "_time": "1557297801",
                                         "length": 4
                                 }
                         }
                 ]
value            0 wei
```

```
currentDeposit
0: uint256: 20

gameIDs
0: uint256: 1557297660

showGameCount
0: uint256: 0
```

Increased honesty deposit

This time, `0xca3...a733c` does not resubmit the secret input. `0x147...c160c` then wins. Note the `currentDeposit` has risen back to 20 wei after the game resolution due to dishonest play.

4 Source Code

The source code for Lottereum can be found on Github.