




360 核心安全技术博客

 主页 Home

 归档 Archive

 分类 Category

 关于 About



Not A Fair Game – Fairness Analysis of Dice2win

10月12, 2018

Zhiniang Peng
from Qihoo
360 Core
Security

Dice2win is currently an extremely popular blockchain betting game on Ethereum. Known for its "provable fairness",

Dice2win

currently has a bet of nearly one thousand Ethers (about 1.5 million RMB) per day, being the second largest Ethereum betting game after etheroll. However, our analysis found that all games in dice2win have fairness vulnerabilities, and the dealer can use these vulnerabilities to manipulate game results.

文章目录

- [Introduction to Dice2win](#)
- [Selective Abort Attack in Dice2win](#)
- [Selectively Draw a Lottery Attack](#)
- [Drawing Arbitrary Lottery \(Merkle proof Verification Bypass Vulnerability\)](#)
 - [Analysis on a Merkle proof verification bypass attack that has occurred](#)
 - [Merkle proof verification bypasses vulnerability:](#)
 - [Other security issues:](#)
- [Conclusions](#)




Introduction to Dice2win

Dice2win currently has several games including "Coin flip", "Roll a dice", "Two dices" and "Etherroll".





360 核心安全技术博客

-  主页 Home
-  归档 Archive
-  分类 Category
-  关于 About



In these games, each user bets separately and makes a one-on-one bet with the dealer. The essence of the game is that users and dealers generate random numbers through a series of protocols on the decentralized Ethereum smart contract platform. If the user has the lucky guess on the right random number, the user wins, otherwise the dealer wins.

Before we dive into the Dice2win workflow and fairness, let's discuss a long-standing cryptography problem: Mental poker. Mental poker was first introduced by Shamir, Rivest and Adleman in the article "Is it possible to play a fair game of 'Mental Poker'" in 1978. (Shamir, Rivest and Adleman are familiar? Yes, that's the same RSA you know). The essence is a solution for the problem: in the absence of a trusted third parties (trusted platform or software), how can the two dishonest participants play a fair chess game on the web. In the definition of fairness, there is a very important point: if any party receives the result of the game, then all honest parties should receive the same result.





Dice2win is actually a typical case of using the blockchain to implement mental poker. But we found that Dice2win does not guarantee the fairness and security of mental poker.

Selective Abort Attack in Dice2win

Here let's take a look at how Dice2win works. The essence of the game is that users and dealers generate random numbers through a series of protocols on the decentralized Ethereum smart contract platform. If the user guesses the correct random number, the user wins, otherwise the dealer wins. The overall workflow of the game is as follows:



360 核心安全技术博客

-  主页 Home
-  归档 Archive
-  分类 Category
-  关于 About








1. [Dealer commits] The dealer (secretSigner) randomly generates a random number reveal and calculates $\text{commit} = \text{keccak256}(\text{reveal})$, then the dealer commits to the reveal. According to the current block height, set the last block's height `commitLastBlock` used by the commitment. Sign the combination of `commitLastBlock` and `commit` to get `sig` and send (`commit`, `commitLastBlock`, `sig`) to players.
2. [PlaceBet] After the players get (`commit`, `commitLastBlock`, `sig`), they will select the specific game to play, guess a random number `r`, and send the bet trade `placeBet` to the smart contract to place a bet.
3. [Block generated] The bet transactions are included by miners into `block1`, and the players' bet contents are saved in the storage of the contract.
4. [SettleBet] When the dealer sees the players' bet information in `block1`, the dealer will send the `settleBet` transaction with public commitment value reveal to the blockchain. Then the contract calculates the random number $\text{random_number} = \text{keccak256}(\text{reveal}, \text{block1.hash})$. If `random_number` satisfies the user's betting condition, the user wins, otherwise the dealer wins. In addition, the game also has a jackpot mechanism, that is, if a `random_number` coincides with a certain value (such as 88888), the user can win the jackpot in the bonus pool.

Dice2win claims that its games have mathematically verifiable fairness on their official website and whitepaper. The random number generation process is determined by the miners and the dealers. The miners or dealers can't control the game results, so



360 核心安全技术博客

-  主页 Home
-  归档 Archive
-  分类 Category
-  关于 About
- 

players can bet on the bet with trust. In addition, in some articles elaborating the security of Ethereum smart contract, we also saw some authors refer Dice2win's random number generation process as a best security practice.

However, our analysis found that all games in dice2win can be selectively aborted by the dealer, and the dealer can selectively publish the winning results, which will lead to the user not winning the game or the lottery. Let's see the following two attack scenarios:

Scenario 1:

This is when the user has a large bet amount and the odds are high. After the user places a bet to generate block1, block1.hash is actually fixed. At this point, the dealer can already calculate the random_number to get the user's bet results and profit/loss. The dealer can then selectively abort the transaction. If the user does not win the prize, the dealer will announces the draw result as usual. If the user wins the prize, the dealer may refuse to settle the user's bet (claiming that this is because of network users and technical issues). Then the user's bet becomes invalid.






Scenario 2:

Assuming the user's bet amount is not large, but after the block1 is generated, the dealer finds that the random_number results in the user winning the jackpot. The dealer can then selectively abort the protocol and let the bet become invalid.

In both scenarios, the dealer can easily control the results of the transaction. Of course, the dealer does not launch such an attack on every transaction, but can choose to control the award-winning transaction if the bet is large. Dice2win officially has already stated in the note of the smart contract code that the



360 核心安全技术博客

-  主页 Home
-  归档 Archive
-  分类 Category
-  关于 About
- 

"technical problem and Ethereum congestion" may cause the dealer to fail to draw the prize (about 1 hour), when it happens, the user can withdraw the bet.

```
// This is a check on bet mask overflow.
uint constant MAX_BET_MASK = 2 ** MAX_MASK_MODULO;

// EVM BLOCKHASH opcode can query no further than 256 blocks into the
// past. Given that settleBet uses block hash of placeBet as one of
// complementary entropy sources, we cannot process bets older than this
// threshold. On rare occasions dice2win croupier may fail to invoke
// settleBet in this timespan due to technical issues or extreme Ethereum
// congestion; such bets can be refunded via invoking refundBet.
uint constant BET_EXPIRATION_BLOCKS = 250;
```

The essential reason for this vulnerability is that the random number of the scheme is not really random for the dealer. The dealer can know the result of the bet in advance. Think about it if you go to roll a dice in a casino that claims to be "absolutely fair." Upon the betting, there a way for the dealer to steal a look at the results of the dice. After calculate profit and loss, the dealer decides whether to open or re-roll, will you still call it really random?





In fact, selective abort attack is one of the most common attacks against mental poker fairness. It is actually very simple to fix the problem, as long as there is a penalty mechanism to force the dealer to open the commitment within the time limit. Or we can use some other random number generation (PRG) algorithms for mental poker or Secure Multi-party Computation to replace the PRG here..

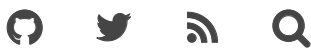
Selectively Draw a Lottery Attack

The fix of the selective abort attack actually can simply require the dealer to open the commitment within a limited time. But this does not completely eliminate Dice2win's advantage from the mechanism. Then does it work by simply and directly introducing other random number generation algorithms for Secure Multi-party Computation into here? In fact, it may still not fully guarantee the fairness of the game.



360 核心安全技术博客

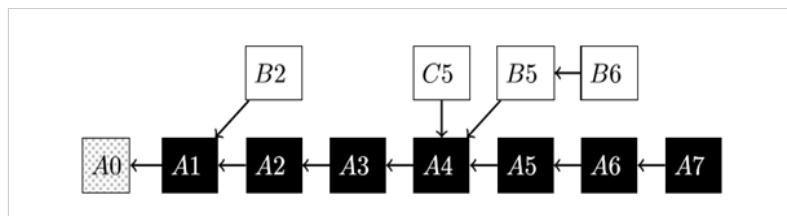
-  主页 Home
-  归档 Archive
-  分类 Category
-  关于 About



Here we would like to introduce one of our interesting findings:

The communication model on the smart contract platform is different from the traditional peer-to-peer communication model on the Internet. The security protocol proved under the traditional peer-to-peer communication model may not be guaranteed to be secure if directly applied to the smart contract platform. The core reason is that under the traditional peer-to-peer communication model, the execution of the protocol is sequential and irreversible, while in the communication model of smart contracts, the implementation of the protocol may be non-sequential, or reversible, due to the possibility of forking in some consensus algorithms such as POW.






In the figure below, assuming that the black blocks form the main chain of the network and the white blocks are the forked blocks. If a step of the Secure Multi-party Computation protocol (certain transaction) is executed in white blocks, the transaction will not take effect. For example, Alice and Bob are executing some protocol on the blockchain. If Alice executes a transaction on block B5 (maybe some payment), and Bob then will announce a secret on block B6. Then, because of the network forking, the transactions on B5 and B6 both failed. But Alice received the secret.



Ethereum uses the "GHOST protocol" to select blocks to become the main chain. The forked blocks include uncle blocks and orphan blocks. As you can see in the figure below, the probability of the current ration of uncle blocks in Ethereum has exceeded 10%.



360 核心安全技术博客

-  主页 Home
-  归档 Archive
-  分类 Category
-  关于 About
- 

Therefore, when directly transplanting a mental poker algorithm or some protocols of Secure Multi-party Computation to the smart contract platform, the probability of unstable events is not negligible.

Ethereum - The Ethereum Blockchain Explorer					
Price	Difficulty	Block time	Hashrate	TPS	Uncle rate
\$228.6 B0.034 +1.3%	3,315 T +0.9%	14.0 s +1.5%	267.4 TH/s +1.3%	6.4 +13.3%	12.2 % +0.9%




The most straightforward way to solve this problem is to wait long enough between each step of the protocol in the smart contract. When it is nearly 100% sure that the previous transaction has stabilized in the block, then execute the next transaction. But in this way, we need to wait multiple blocks (several minutes) for a transaction to be finalized. For gambling games, it is obviously unacceptable.

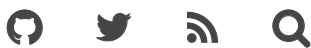
In fact, Dice2win has realized the problems caused by the uncle blocks last month and claimed that the MerkleProof algorithm proposed by them can solve the problem and make the game fair. In the commit, the method of MerkleProof was introduced to settle a bet in the uncle block (<https://github.com/dice2win/contracts/commit/86217b39e7d069636b04429507c64dc061262d9c>).

Now let's see how Dice2win solves this problem. In the Dice2win code (<https://github.com/dice2win/contracts/blob/b0a0412f0301623dc3af2743dcace8e86cc6036b/Dice2Win.sol>), we can see the method `settleBetUncleMerkleProof(uint reveal, uint40 canonicalBlockNumber)`:



360 核心安全技术博客

-  主页 Home
-  归档 Archive
-  分类 Category
-  关于 About



```
function settleBetUncleMerkleProof(uint reveal, uint40 canonicalBlockNumber) external onlyCroupier {
    // "commit" for bet settlement can only be obtained by hashing a "reveal".
    uint commit = uint(keccak256(abi.encodePacked(reveal)));

    Bet storage bet = bets[commit];

    // Check that canonical block hash can still be verified.
    require (block.number <= canonicalBlockNumber + BET_EXPIRATION_BLOCKS, "Blockhash can't be queried by EVM.");

    // Verify placeBet receipt.
    requireCorrectReceipt(% + 32 + 32 + 4);
}
```

The core logic of the MerkleProof algorithm proposed by Dice2win is that in order to improve the user experience (quick draw), when the dealer receives a bet transaction (Tx) (assuming it is in block B5), he settle the bet as soon as possible. However, due to the GHOST algorithm, the network may chose A5 as the block on the main chain (the hash value is different from B5, so the lottery result is different). At this time, according to the original SettleBet method of the contract, it is impossible to SettleBet for the block B5. For this problem, Dice2win's solution is: directly SettleBet to the uncle block.






How to SettleBet for the uncle block: because the hash of the uncle block is contained in a block on the main chain. There are many hash correlation structures in the block structure of Ethereum. Specifically, we can look at the definition of the Ethereum block header:

```
// Header represents a block header in the Ethereum blockchain.
type Header struct {
    ParentHash common.Hash    `json:"parentHash"    gencodec:"required"`
    UncleHash  common.Hash    `json:"sha3Uncles"    gencodec:"required"`
    Coinbase   common.Address `json:"miner"         gencodec:"required"`
    Root       common.Hash    `json:"stateRoot"     gencodec:"required"`
    TxHash     common.Hash    `json:"transactionsRoot" gencodec:"required"`
    ReceiptHash common.Hash    `json:"receiptRoot"   gencodec:"required"`
}
```

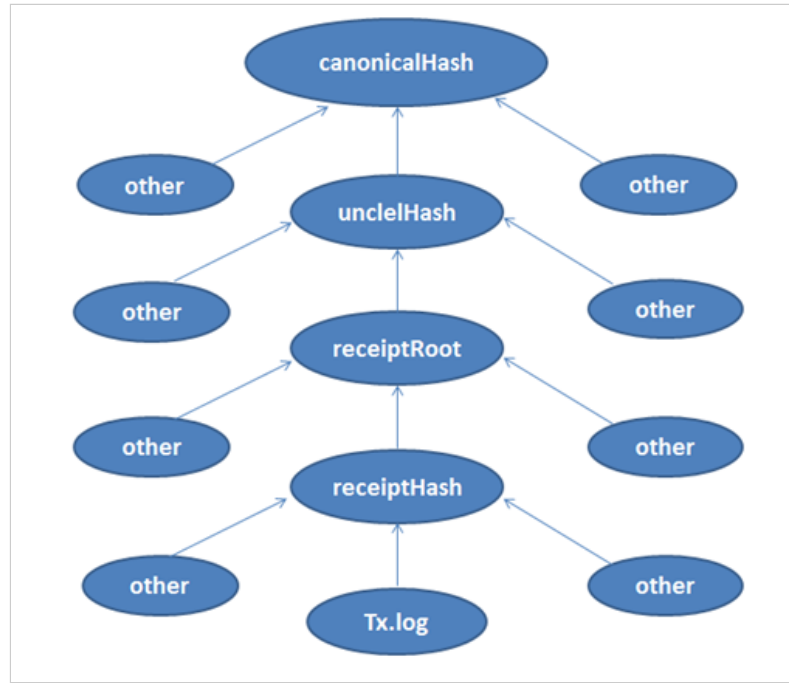
So from execution result ReceiptHash of trade Tx in B5, we can always calculate the hash value to get the receiptsRoot of the uncle block B5. Then hash with



360 核心安全技术博客

-  主页 Home
-  归档 Archive
-  分类 Category
-  关于 About
- 





other structures to get the block hash of B5 (we call it uncleHash). Assuming that the uncleHash of the unblock B5 is referenced in the block A6, then we can finally construct a non-structural Merkle Tree with the canonicalHash of A6 being the root node. Transaction Tx is one of the leaf nodes. Its structure is as follows:



The idea of the Merkle proof algorithm proposed by Dice2win is that when the fork block is generated (such as B5 and A5 at the same time), the network may have two lottery results. But due to network issues, the dealer may receive one of the blocks first (assumed to be B5). From the perspective of the dealer, it does not know whether B5 will be a block on the main chain. Even if the dealer waits for a while and receives (A5, A6, B6), it still cannot determine which chain will be the main chain. In order to increase the speed of the draw, when the dealer receives a transaction block, the draw is immediately made. If this transaction block becomes the main chain (A5), the SettleBet method is used as usual to draw the prize. If the block ends up as an uncle block (B5), then submit a proof of existence of the unstructured Merkle tree (as shown above) with this transaction result as the leaf node and the main block canonicalHash



360 核心安全技术博客

-  主页 Home
-  归档 Archive
-  分类 Category
-  关于 About



referencing this uncle block as the root node. This proves that B5 does exist, and the transaction Tx is included in B5; the dealer can use the uncle block to draw the prize.

Dice2win's Merkle proof algorithm seems to be a good idea to solve the speed issue of the decentralized gambling games on Ethereum. But in fact, this is not fair. The fact that nearly 10% blocks are uncle blocks in Ethereum can lead to the dealers' advantage to make selective draws based on the results of the game. If the dealer A5 wins, he will open A5. If the B5 dealer wins, he will open B5. Such a game is obviously unfair.

Drawing Arbitrary Lottery (Merkle proof Verification Bypass Vulnerability)

After reading Dice2win's implementation of the Merkle proof in detail, we found that there are many bypass methods for the non-structural Merkle Proof verification of the contract. That is, the dealer can forge a Merkle proof of the uncle B5, and deceive the contract to draw the prize for any result.

Analysis on a Merkle proof verification bypass attack that has occurred

When we looked through the history of the contract and found that, in fact, last month, an attacker had already bypassed the Merkle proof algorithm, and looted every penny of the contract balance (but the situation didn't draw the attention of the community, Dice2win officially processed the incident silently). Before we introduce the vulnerability we found, we can take a look at the attack that has occurred on the Merkle proof verification algorithm. One of the attack




360 核心安全技术博客

 主页 Home

 归档 Archive

 分类 Category

 关于 About

transactions occurred here:

<https://etherscan.io/tx/0xd3b1069b63c1393b160c65481bd48c77f1d6f2b9f4bde0fe74627e42a4fc8f81>

TxHash:	0xd3b1069b63c1393b160c65481bd48c77f1d6f2b9f4bde0fe74627e42a4fc8f81
TxReceipt Status:	Success
Block Height:	6285180 (202246 Block Confirmations)
TimeStamp:	33 days 6 hrs ago (Sep-07-2018 12:06:01 AM +UTC)
From:	0x54b7eb670e091411f82f50fdee3743bd03384aff

The attacker automatically waged a bet by creating an attack contract

0xc423379e42bb79167c110f4ac541c1e7c7f663d8

and called the placeBet method on contract

0xc423379e42bb79167c110f4ac541c1e7c7f663d8

(17 bets, 2 ethers at a time). Then forged the Merkle proof, called the settleBetUncleMerkleProof method to draw the prize. After winning 33 Ethers, transfer the bonus to the account









0x54b7eb670e091411f82f50fdee3743bd03384aff, and

in the end, the contract was self-destructed. Through the reverse analysis of the contract bytecode, we can know that the attack exploited the following vulnerabilities:

1. In Dice2win different versions of the contract, there may be same secretSigner, which will result the commitment of the dealer be used in different versions of contracts. *[Security vulnerabilities caused by operation and maintenance]*
2. The expiration check for the commit in the placeBet method can be bypassed. The commitLastBlock is uint256 type when it is



360 核心安全技术博客

-  主页 Home
 -  归档 Archive
 -  分类 Category
 -  关于 About
-    

checked with the current block.number. Then when it is brought into keccak256 for signature verification, it is converted to uint40. Then the attacker can change the highest bit (256bit) of any commitLastBlock signed by the secretSigner from 0 to 1, which can bypass the time verification.

_[this vulnerability has not been fixed in the latest version, see the figure below] _

```
function placeBet(uint betMask, uint modulo, uint commitLastBlock, uint commit, bytes32 r, bytes32 s) external payable {
```

3. Merkle proof verification is not strict. In the old version of settleBetUncleMerkleProof, the boundary check for each calculation of the hashSlot offset is not strict, which causes the attacker to bypass the validation without binding the target commit to the calculation of the Merkle proof. _ [This vulnerability has been fixed, see the figure below]_





```
function settleBetUncleMerkleProof(uint reveal, uint40 canonicalBlockNumber) external {
```

Merkle proof verification bypasses vulnerability:

After our analysis, we find that there are still a variety of bypass methods in Dice2win's current version. Since this method can only be called by the dealer at present, so common attackers cannot exploit this



360 核心安全技术博客

-  主页 Home
-  归档 Archive
-  分类 Category
-  关于 About



vulnerability. But the vulnerability can be used as backdoor by the dealer to achieve arbitrary draws. Here we roughly tidy up the validation logic of the current verification algorithm:

1. First call the `requireCorrectReceipt` method to verify that the Receipt format satisfies the condition.
2. The Receipt entry contains a successful transaction.
3. The target address of the transaction is the Dice2win contract.
4. The starting leaf node of the Merkle Proof validation calculation contains the target commit.
5. The calculated `canonicalHash` is a legal main block hash.

The satisfaction of conditions 1, 2, and 3 is easy to achieve; we only need to construct a data format that satisfies the condition. Condition 4 and 5 is essentially an iterative calculation: $\text{hash}_0 = \text{commit}$ $\text{hash}_{\{n+1\}} = \text{SHA3}(\text{something}_{\{n1\}}, \text{hash}_n, \text{something}_{\{n2\}})$ $\text{canonicalHash} = \text{hash}_{\{\text{lastone}\}}$

Attack Method 1:

It is not difficult to put a target commit into an uncle block transaction. The dealer can inject the commit into the input data of some transaction, if the transaction is included in an uncle block, dealer can use it to construct a fake Merkle proof.

Attack Method 2:

Since the depth of the iterative calculation of $\text{hash}_{\{n+1\}} = \text{SHA3}(\text{something}_{\{n1\}}, \text{hash}_n, \text{something}_{\{n2\}})$ is not checked, so the dealer can generate a new Merkle tree locally. The leaf nodes of this Merkle tree satisfy conditions 1, 2, and 3 and contain multiple `commit_i`. By embedding the root





360 核心安全技术博客

 主页 Home

 归档 Archive

 分类 Category

 关于 About

hash of the Merkle tree into a normal block, a fake proof can be generated. In this attack method, the dealer can draw an arbitrary lottery for all commits once and for all.

The core problem with these bypass methods is that the unstructured Merkle tree does not actually satisfy the structure of the Merkle hash tree we know. The conventional Merkle hash tree can be used to prove existence of something, but it is difficult for Dice2win's unstructured Merkle proof algorithm to achieve this goal.

Other security issues:

When the user bets are not drawn, the user can call refundBet to overflow jackpotSize, causing jackpotSize to become a huge integer (discovered and disclosed by Gaia).





```
function refundBet(uint commit) external {  
    // Check that bet is in 'active' state.  
    Bet storage bet = bets[commit];  
    uint amount = bet.amount;  
    require (amount != 0, "Bet should be in an 'active' state");  
}
```

Conclusions

1. Dice2win is not a fair betting game.
2. The security issue of smart contracts is very severe. (This is actually the first smart contract I analyzed)
3. Sometimes, traditional protocols for Secure Multiparty Computation cannot be directly applied to smart contract because of the differences in their communication models.



360 核心安全技术博客

-  主页 Home
-  归档 Archive
-  分类 Category
-  关于 About



本文链接:

http://blogs.360.cn/post/Fairness_Analysis_of_Dice2win_EN.html

-- EOF --

作者 [admin001](#) 发表于 2018-10-12 05:49:45, 添加在分类 [Blockchain Vulnerability Analysis](#) 下, 最后修改于 2018-10-12 06:16:19

分享到: [新浪微博](#) [微信](#) [Twitter](#) [印象笔记](#) [QQ好友](#) [有道云笔记](#)

« [剪贴板幽灵：币圈的神偷圣手](#)
[Not a fair game, Dice2win公平性分析](#) »

Comments

© 2020 - 360 核心安全技术博客 - blogs.360.cn

Powered by [ThinkJS](#) & [FireKylin 1.2.8](#)