buu练题记录15-[ACTF新生赛2020]SoulLike

原创

Asteri5m 于 2021-06-21 21:22:46 发布 97 收藏

分类专栏： Reserve buuctf

本文链接：https://blog.csdn.net/qq_45892237/article/details/118094978


Reserve 同时被 2 个专栏收录

18 篇文章 1 订阅

订阅专栏


buuctf

16 篇文章 0 订阅

订阅专栏

首先拿到查壳，发现是64位的ELF文件，所以直接IDA打开，找到main函数

```
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  char v3; // al
  __int64 result; // rax
  char v5; // [rsp+7h] [rbp-B9h]
  signed int i; // [rsp+8h] [rbp-B8h]
  signed int j; // [rsp+Ch] [rbp-B4h]
  int v8[14]; // [rsp+10h] [rbp-B0h]
  int v9; // [rsp+48h] [rbp-76h]
  __int16 v10; // [rsp+4Eh] [rbp-72h]
  char v11[17]; // [rsp+50h] [rbp-70h]
  char v12; // [rsp+61h] [rbp-5Fh]
  unsigned __int64 v13; // [rsp+B8h] [rbp-8h]

  v13 = __readfsqword(0x28u);
  printf("input flag:", a2, a3);
  scanf("%s", v11);
  v9 = 'ftca';                                  // actf
  v10 = '{';
  v5 = 1;
  for ( i = 0; i <= 4; ++i )
  {
    if ( *(&v9 + i) != v11[i] )                 // 检验头字符串是否为 actf{
    {
      v5 = 0;
      goto LABEL_6;
    }
  }
  if ( !v5 )
    goto LABEL_19;
LABEL_6:
  for ( j = 0; j <= 11; ++j )
    v8[j] = v11[j + 5];                         // 将{}内内容赋值给v8
  v3 = sub_83A(v8) && v12 == '}' ? 1 : 0;       // len = 18
  if ( v3 )
  {
    printf("That's true! flag is %s", v11);
    result = 0LL;
  }
  else
  {
LABEL_19:
    printf("Try another time...");
    result = 0LL;
  }
  return result;
}
```

- v9是这里解释为int类型，但是可以转为字符，所以为小端存储，既倒着取值；
- 这里效验v12为字符“}”，但是我们的输入在v11，所以输入应该为18位长度的字符串，刚好把最后一位溢出到v12；
- 这里只调用一个sub_83A函数，所以直接查看这个函数。但是F5的时候IDA会提示 `too big function`，解决办法为修改IDA配置文件*cfg\hexrays.cfg*

```
MAX_FUNCSIZE            = 64        // Functions over 64K are not decompiled

// 修改为:
MAX_FUNCSIZE            = 1024        // Functions over 64K are not decompiled
```

然后这里反编译的时候较长，去刷会抖音再回来，就看到

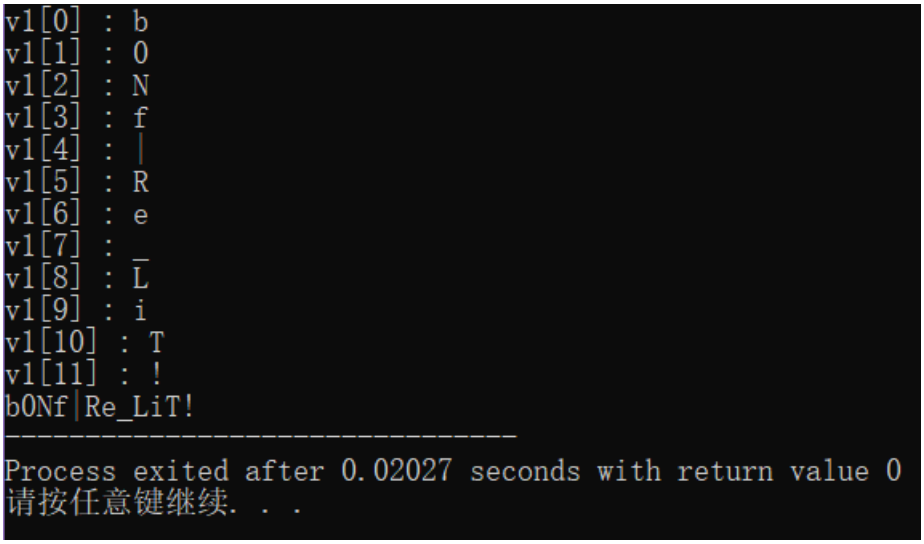
```
3018  v1[9]  = 0x70u;
3019  v1[10] ^= 0x29u;
3020  v1[11] ^= 0x3Bu;                          // 250  轮异或
3021  v4 = 126;
3022  v5 = 50;
3023  v6 = 37;
3024  v7 = 88;
3025  v8 = 89;
3026  v9 = 107;
3027  v10 = 53;
3028  v11 = 110;
3029  v12 = 0;
3030  v13 = 19;
3031  v14 = 30;
3032  v15 = 56;
3033  for ( i = 0; i <= 11; ++i )
3034  {
3035    if ( *(&v4 + i) != a1[i] )              // 效验
3036    {
3037      printf("wrong on #%d\n", i);          // 会给出哪一步错误了
3038      return 0LL;
3039    }
3040  }
3041  return 1LL;
3042}
```

3000行代码，250轮xor操作（其实有些不是，是++），这里效验出错的时候，它会提示你是哪一步错了，所以这里应该可以从头开始逐位爆破。

这里我写了一个C的exp：

```c
#include<stdio.h>

void sub_83A(int *v1){
 v1[0] ^= 0x2B;
 v1[1] ^= 0x6C;
 v1[2] ^= 0x7E;
 v1[3] ^= 0x56;
 v1[4] ^= 0x39;
 v1[5] ^= 3;
 v1[6] ^= 0x2D;
 v1[7] ^= 0x28;
 v1[8] ^= 8;
 ++v1[9];
 v1[10] ^= 0x2F;
 v1[11] ^= 0xA;
    ........  //太多不宜展示，就仅复制250轮加密就OK
}

int main(){
 int a[] = {126,50,37,88,89,107,53,110,0,19,30,56,}; //效验值

 int v1[20];  //存放临时值
 int flag[20];  //存放flag

 for(int x=0;x<=11;){
  for(int y=0;y<=125;y++){
   for(int z=0;z<x;z++)
    v1[z] = flag[z];  //因为每次v1整个数组的值都要变,所以重新赋值正确值
   v1[x] = y;
   sub_83A(v1);  //经过函数
   if(v1[x] == a[x]){
    flag[x] = y; //正确值进行存储
    printf("v1[%d] : %c\n",x,y);  //显示进度
    x ++;
    y = 0;
   }
  }
  if(x < 12){
   printf("ERROR : %d",x);  //出错了就提示
   return 0;
  }
 }

 for(int i = 0;i<=11;i++){
  printf("%c",flag[i]); //输出完整flag
 }

 return 0;
}
```

成功获取flag，因为是在buu上刷题，所以包上flag{}即可。



爆出flag后，想去看了看其他师傅的解，发现有也可以用python写，使用pwntools库。纯Re选手可能不清楚，可以了解一下。