# BUUCTF——[ACTF新生赛2020]SoulLike——使用angr解

原创

1ens 于 2021-11-16 20:17:19 发布 
分类专栏： BUU刷题 文章标签： 安全



64位无壳。IDApro打开，查看main函数

```
__int64 __fastcall main(int a1, char **a2, char **a3)
{
  char v5; // [rsp+7h] [rbp-B9h]
  int i; // [rsp+8h] [rbp-B8h]
  int j; // [rsp+Ch] [rbp-B4h]
  int v8[14]; // [rsp+10h] [rbp-B0h] BYREF
  char v9[110]; // [rsp+4Ah] [rbp-76h] BYREF
  unsigned __int64 v10; // [rsp+B8h] [rbp-8h]

  v10 = __readfsqword(0x28u);
  printf("input flag:");
  scanf("%s", &v9[6]);                          // 把读取存在a[6]以后
  strcpy(v9, "actf{");
  v5 = 1;
  for ( i = 0; i <= 4; ++i )                    // 验证前五位
  {
    if ( v9[i] != v9[i + 6] )
    {
      v5 = 0;
      goto LABEL_6;
    }
  }
  if ( !v5 )
    goto LABEL_16;
LABEL_6:
  for ( j = 0; j <= 11; ++j )                   // 把输入的东西赋值给v8
    v8[j] = v9[j + 11];
  if ( (unsigned __int8)sub_83A(v8) && v9[23] == '}' )// 验证最后一位
  {
    printf("That's true! flag is %s", &v9[6]);
    return 0LL;
  }
  else
  {
LABEL_16:
    printf("Try another time...");
    return 0LL;
  }
}
```



逻辑很简单，我们来查看sub_83A函数。点进去，toobig

看了其他博客。

我们需要将ida /ctg目录下的hexrays.cfg文件中的MAX_FUNCSIZE=64 改为 MAX_FUNCSIZE=1024

```
MAX_FUNCSIZE            = 1024        // Functions over 64K are not decompiled
MAX_FUNC_ARGS           = 64        // Max number of function arguments
```



改完看函数

好长：看最后部分，一大堆异或，然后比较

```
for ( i = 0; i <= 11; ++i )
{
  if ( v3[i] != a1[i] )
  {
    printf("wrong on #%d\n", (unsigned int)i);
    return 0LL;
  }
}
return 1LL;
}
```



第一个想到了爆破，最近在学angr，尝试用angr解题

**脚本：**

```
import angr
p = angr.Project('SoulLike',load_options={"auto_load_libs":False})
st = p.factory.entry_state()
sm = p.factory.simulation_manager(st)
sm.explore(find=0x411176, avoid=0x411195)
print(sm.found[0].posix.dumps(0))
```

注意：载入之后会有两个warning，第一个提示注意偏移地址，要加上0x400000

第二个是线程局部存储，没搞懂，不管他；

```
>>> p = angr.Project('SoulLike')
WARNING | 2021-11-16 03:34:47,659 | cle.loader | The main binary is a position-i
ndependent executable. It is being loaded with a base address of 0x400000.
WARNING | 2021-11-16 03:34:47,684 | cle.backends.tls | The provided object has a
ftware alid tls_data_size. Skip TLS loading.
```



find的地址：也就是判断为flag正确的地址

```
.text:0000000000011176 lea     rax, [rbp+var_76+6]
.text:000000000001117A mov     rsi, rax
.text:000000000001117D lea     rdi, aThatSTrueFlagI ; "That's true! flag is %s"
.text:0000000000011184 mov     eax, 0
.text:0000000000011189 call    _printf
.text:000000000001118E mov     eax, 0
.text:0000000000011193 jmp     short loc_111AB
```



void的地址：也就是flag判断错误的地址

```
.text:0000000000011195
.text:0000000000011195 loc_11195:
.text:0000000000011195 lea     rdi, aTryAnotherTime ; "Try another time..."
.text:000000000001119C mov     eax, 0
.text:00000000000111A1 call    _printf
.text:00000000000111A6 mov     eax, 0
```



最后的flag：

actf{b0Nf|Re_LiT!}

```
>>> print(sm.found[0].p
b'actf{b0Nf|Re_LiT!}\x0
```