

# 攻防世界 level0

原创

b0ck 于 2021-05-15 16:45:04 发布 498 收藏

分类专栏: [攻防世界pwn新手区](#) 文章标签: [ubuntu pwn](#) [信息安全](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: [https://blog.csdn.net/weixin\\_45846085/article/details/116853943](https://blog.csdn.net/weixin_45846085/article/details/116853943)

版权



[攻防世界pwn新手区](#) 专栏收录该内容

3 篇文章 0 订阅

订阅专栏

攻防世界第一题, 很简单一个栈溢出。

首先拿到一个elf文件, 把它放到ubuntu中检查一下

```
giantbranch@ubuntu: ~/Desktop/gfsj
giantbranch@ubuntu:~$ cd Desktop/
giantbranch@ubuntu:~/Desktop$ mkdir gfsj
giantbranch@ubuntu:~/Desktop$ cd gfsj/
giantbranch@ubuntu:~/Desktop/gfsj$ mkdir level0
mkdir: cannot create directory 'level0': File exists
giantbranch@ubuntu:~/Desktop/gfsj$ mkdir level2
giantbranch@ubuntu:~/Desktop/gfsj$ ls
level0 level2
giantbranch@ubuntu:~/Desktop/gfsj$ file level0
level0: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked,
interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=8d
c0b3ec5a7b489e61a71bc1afa7974135b0d3d4, not stripped
giantbranch@ubuntu:~/Desktop/gfsj$ checksec level0
[*] '/home/giantbranch/Desktop/gfsj/level0'
Arch:      amd64-64-little
RELRO:     No RELRO
Stack:     No canary found
NX:        NX enabled
PIE:       No PIE (0x400000)
giantbranch@ubuntu:~/Desktop/gfsj$
```

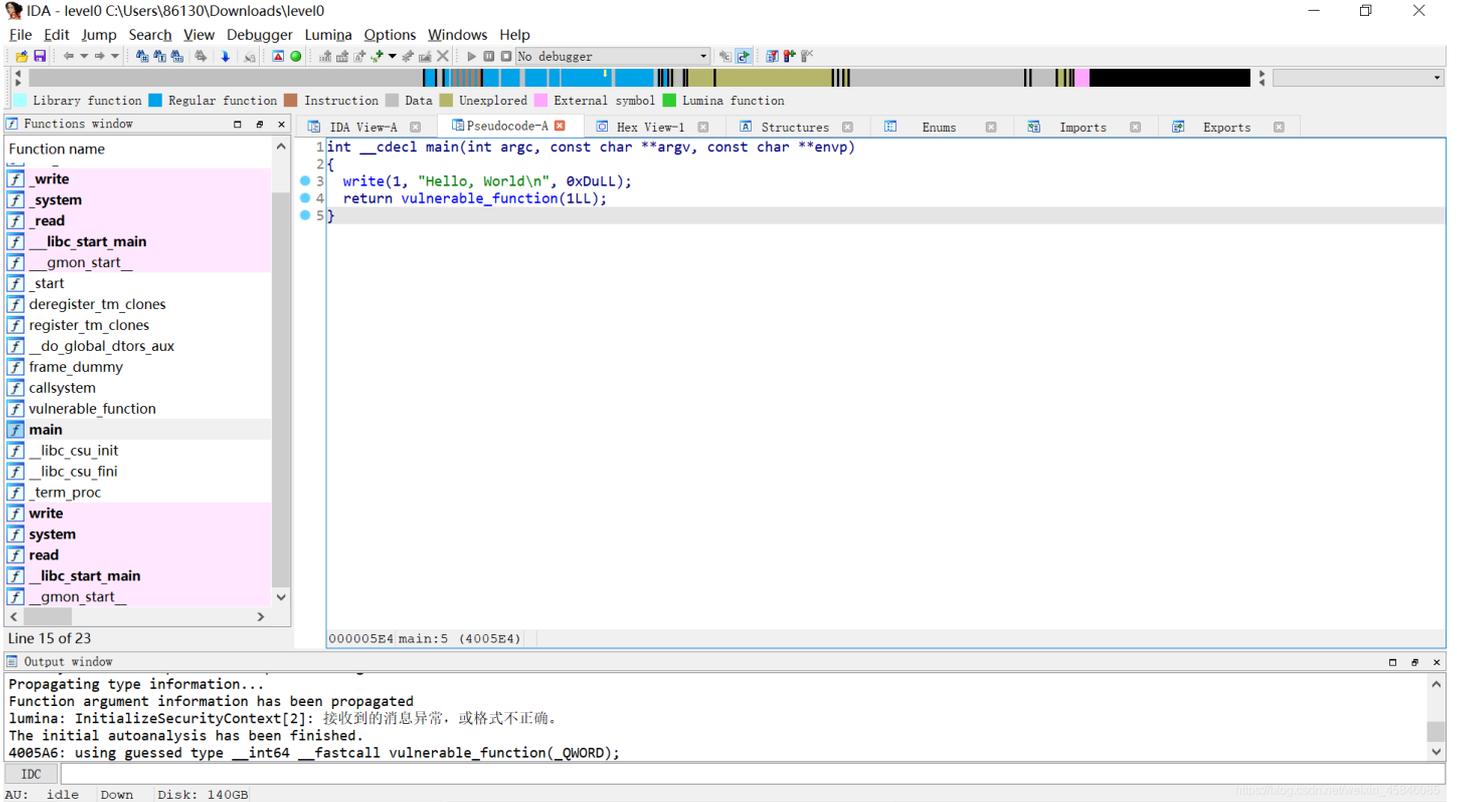
[https://blog.csdn.net/weixin\\_45846085](https://blog.csdn.net/weixin_45846085)

动态链接, 小端序, 64位只打开了NX保护, 接下来看一下文件中引用的字符串

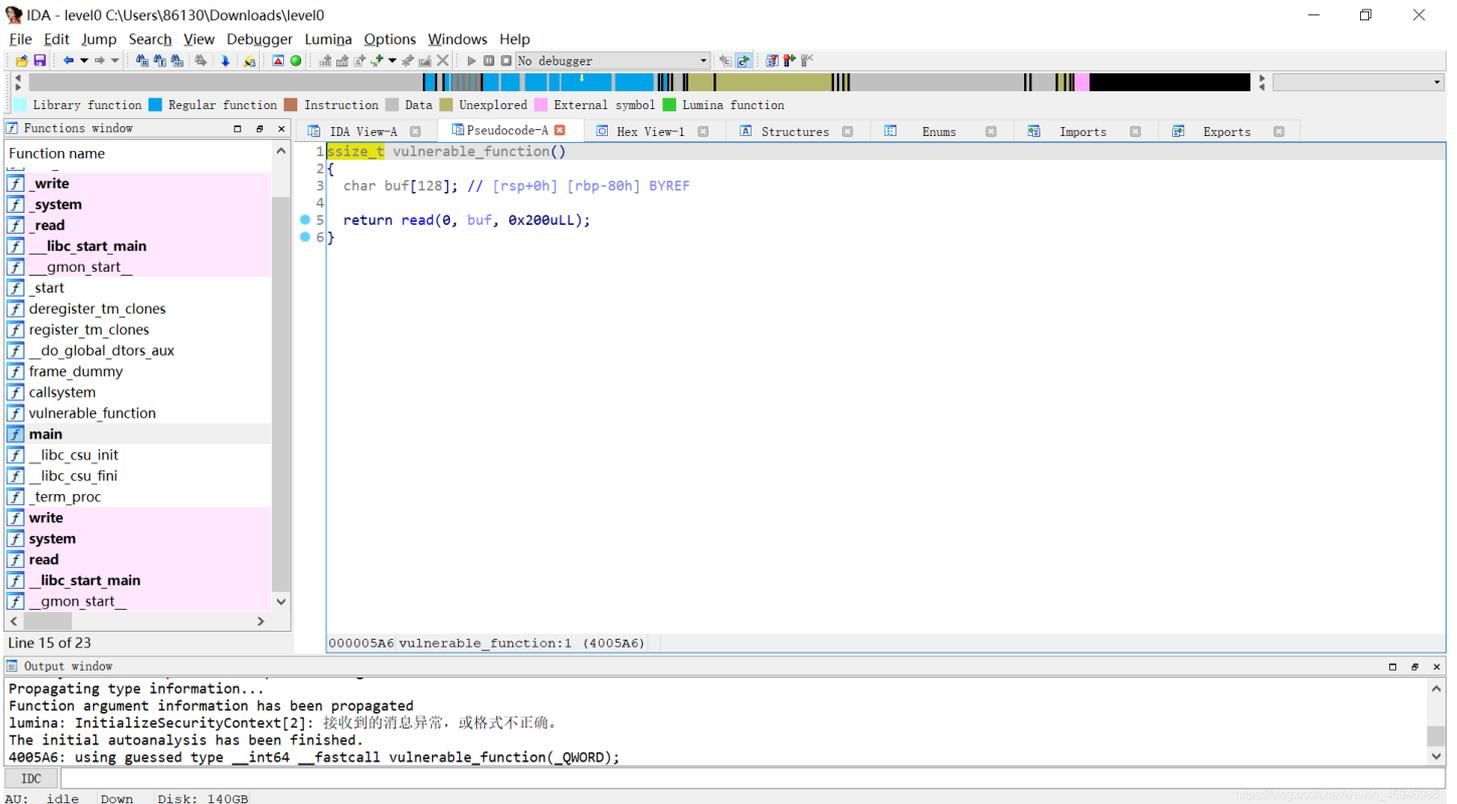
```
strings: 'level0': No such file
giantbranch@ubuntu:~/Desktop$ cd gfsj/
giantbranch@ubuntu:~/Desktop/gfsj$ strings level0
/lib64/ld-linux-x86-64.so.2
libc.so.6
read
system
__libc_start_main
write
__gmon_start__
GLIBC_2.2.5
UH-X
fffff
[]A\A]A^A_
/bin/sh
Hello, World
!*3$
GCC: (Debian 4.9.2-10) 4.9.2
GCC: (Debian 4.8.4-1) 4.8.4
.symbtab
.strtab
.shstrtab
.interp
.note.ABI-tag
.note.gnu.build-id
.gnu.hash
.dynsym
.dynstr
.gnu.version
.gnu.version_r
.rela.dyn
.rela.plt
.init
.text
.fini
.rodata
.eh_frame_hdr
.eh_frame
.init_array
```



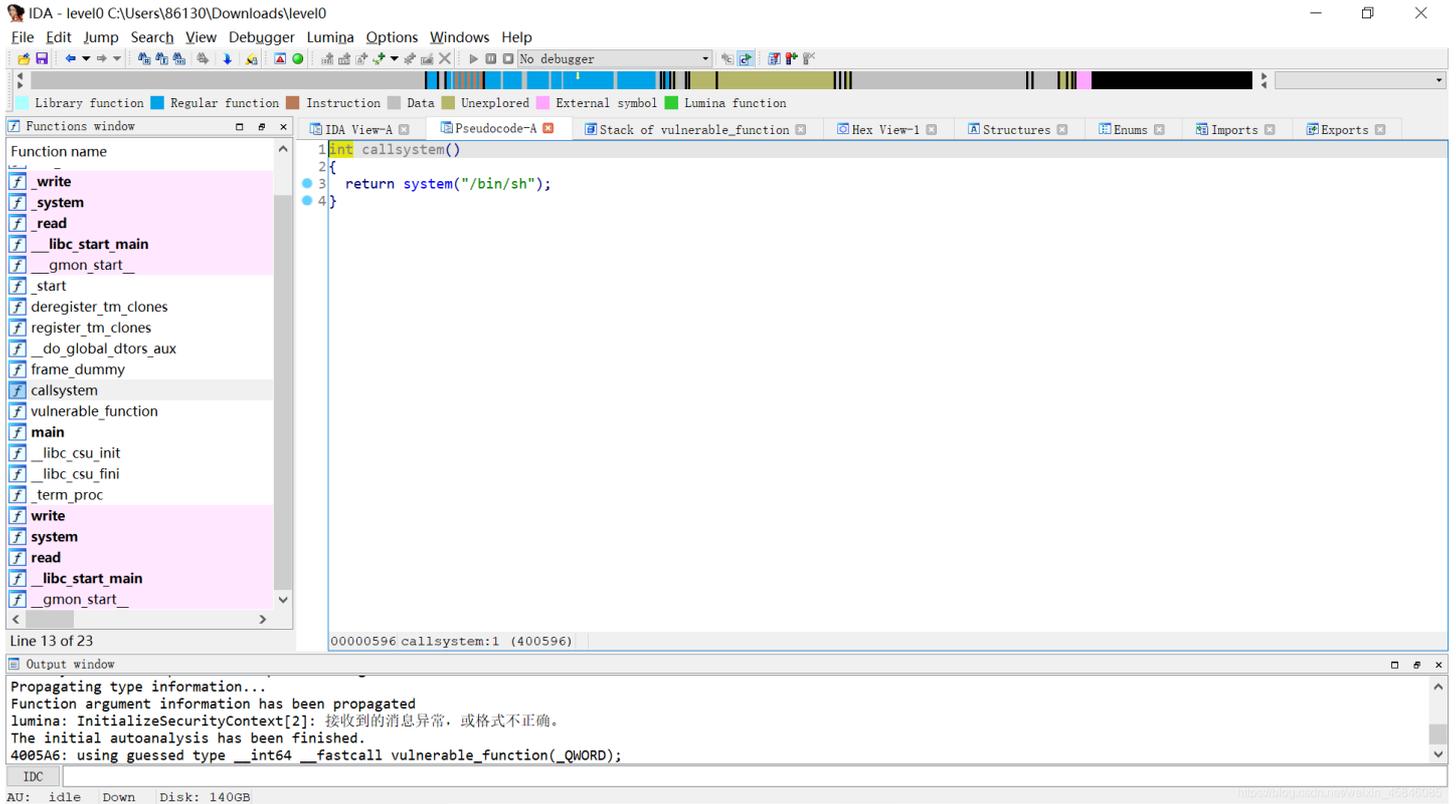
发现有system和/bin/sh，使用64位的IDA反编译一下。



只有一个脆弱函数，很明显了。

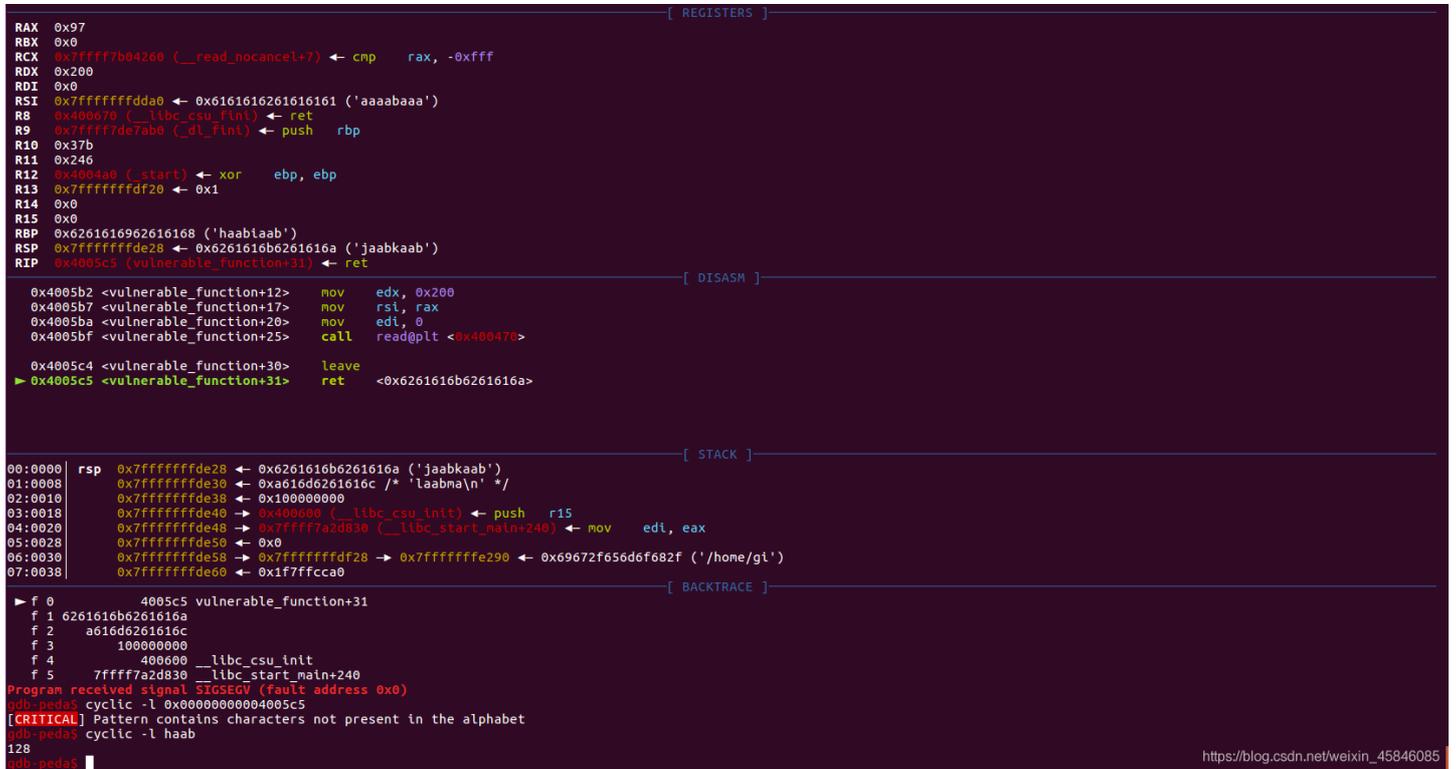


read出存在栈溢出漏洞，接下来寻找一下system和/bin/sh



发现存在后门函数，我们之接利用栈溢出漏洞控制程序跳转到callsystem处就可以getshell了。

gdb调试得到需要填充的垃圾数据



接下来看exp

