

## 운영체제 - 숙제 5, 힙 영역에 코드 넣기

2013210111 남세현

```
[skatpgusskat@localhost Desktop]$ ./heap_main
Start Address Of Heap Area Is 0x8a8e008
Start Address of heap_code() is 0x8a8e03c
return value : DEAD
```

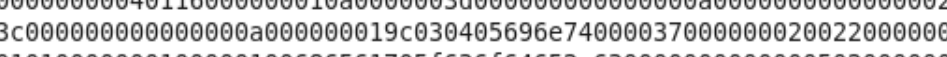
Figure 1 - 실행 결과

숙제의 재미를 더하기 위해서, 7777의 값을 return하지 않고, 0xDEAD 값을 return하도록 하였습니다.

```
int heap_code(void)
{
    return 0xDEAD;
}
```

위 코드를 heap\_code.c에 작성하고, gcc로 오브젝트 파일로 컴파일 하면 아래와 같은 heap\_code.o 파일이 나옵니다.

```
[skatpgusskat@localhost Desktop]$ gcc -c -g heap_code.c  
[skatpgusskat@localhost Desktop]$ perl -e 'local $/; print unpack "H*", <>' heap_code.o
```



The screenshot shows a terminal window with a dark background. The prompt is [skatpgusskat@localhost Desktop]. The user has entered two commands: gcc -c -g heap\_code.c and perl -e 'local \$/; print unpack "H\*", <>' heap\_code.o. The output of the second command is a single line of hexadecimal text.

```
7f454c460101010000000000000000001000300010000000000000000000000a802000000000000  
3400000000002800150012005589e5b8adde00005dc30000011101250e130b030elb0e1101120110  
060000022e003f0c030e3a0b3b0b270c491311011201400a00000324000b0b3e0b03080000004000  
00000300000000000040116000000010a0000003d00000000000000a00000000000000201000000  
000101013c0000000000000000a000000019c030405696e7400003700000000200220000000101fb0e  
0d00010101010100000000100000100686561705f636f64652e630000000000000050200000000133d59  
020200010111c00000000200000000004400000025000000686561705f636f646500000000001c0000
```

오브젝트 파일은 바이너리 데이터인데, 함수의 시작을 바로 알 수 없습니다.

그러므로 Objdump를 이용해서 역어셈블링을 하면

```
[skatpgusskat@localhost Desktop]$ objdump -S heap_code.o
heap_code.o:          file format elf32-i386

Disassembly of section .text:

00000000 <heap_code>:
int heap_code(void)
{
    0:   55                push    %ebp
    1:   89 e5            mov     %esp,%ebp
        return 0xDEAD;
    3:   b8 ad de 00 00  mov     $0xdead,%eax
}

    8:   5d                pop     %ebp
    9:   c3              ret
```

위 결과창에 나온 것 처럼, 처음 함수의 instruction들이 55, 89, e5로 시작됨을 알 수 있습니다.

저 바이너리 데이터에서 55, 89, e5가 나오는 시점을 확인한 후, 거기까지의 offset을 확인합니다.

[illegible]

빨강색을 줄 쳐진 곳에 55, 89, e5가 확인됩니다. 그 앞으로 기타 데이터가 총 52Byte가 있었습니다.

그러므로, 최종적으로 실행한 코드 Heap\_main.c 에서는

1. Heap\_code.o를 읽어서 힙 영역에 복사해놓고
2. 함수포인터를 그 힙 영역의 첫 부분 + offset(52Byte) 를 가리키도록 셋팅한 후 함수포인터로 함수를 실행

```

#include <stdio.h>
#include <sys/mman.h>
#include <stdlib.h>

#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

int main(void)
{
    int fd;                // file discripter written code
    char* heapCodeArea;    // Heap Area for Copy Code
    int (*functionInHeap)(void); // function pointer to execute.
    int ret;               // return value of function

    heapCodeArea = malloc(4096);

    fd = open("./heap_code.o", O_RDONLY);
    if ( fd == 0 )
    {
        printf("File Open Failed\n");
        exit(1);
    }

    read(fd, heapCodeArea, 1024);
    printf("Start Address Of Heap Area Is %p\n",heapCodeArea);

    functionInHeap = heapCodeArea + 0x34;
    printf("Start Address of heap_code() is %p\n",functionInHeap);

    ret = (*functionInHeap)();
    printf("return value : %X\n",ret);

    close(fd);
    free(heapCodeArea);

    return 0;
}

```

숙제를 하면서 어려웠던 점은 없었습니다.