# Understandin Buffer Overflow

## Global Code

### Student.h

**struct** Student {  
 **char** name[12];  
 **char** class[12];  
 **char** gpa[4];  
 **char** student\_id[8];  
} **typedef** Student\_t;  
  
**void** student\_to\_string(**struct** Student s);

### Student.c

#include **<stdlib.h>**#include **<stdio.h>**#include **"student.h"  
  
void** student\_to\_string(**struct** Student s){  
 printf(**"\nName: %s \nClass: %s \nGPA: %s \nID#: %s"**,  
 s.name, s.class, s.gpa, s.student\_id);  
}

## Program 1: Stack Overflow

### Main\_Program.c

#include **<stdlib.h>**#include **<stdio.h>**#include **"student.h"**#include **"Part1.h"**#include **"Part2.h"**Student\_t s;  
  
**int** main(**int** argc, **const char**\* argv[])  
{  
 printf(**"\nStudent Creation Begin\n"**);  
 s = create\_student\_bo();  
 *//s = create\_student\_ho();* student\_to\_string(s);  
}

### Part1.c

#include **<stdio.h>**#include **<stdlib.h>**#include **<memory.h>**#include **"Part1.h"**#include **"student.h"**Student\_t s;  
  
**struct** Student create\_student\_bo(){  
 **char** name[12], class[12], gpa[4], student\_id[8];  
 printf(**"Please input student name:\n"**);  
 gets(name);  
  
 printf(**"Please input student's class [Freshman, Sophomore, Junior, Senior]:\n"**);  
 fgets(class, **sizeof** class, **stdin**);  
 fflush(**stdin**);  
 **if**( strcmp(class, **"Freshman"**) < 0 &&  
 strcmp(class,**"Sophomore"**) < 0 &&  
 strcmp(class,**"Junior"**) < 0 &&  
 strcmp(class,**"Senior"**) < 0)  
 {  
 printf(**"Error, class must equal one of the following [Freshman, Sophomore, Junior, Senior] \n"**);  
 exit(1);  
 }  
  
 printf(**"Please input the student's gpa:\n"**);  
 fgets(gpa, **sizeof** gpa, **stdin**);  
 fflush(**stdin**);  
 **if**(strlen(gpa) > 4 || strlen(gpa) < 0)  
 {  
 printf(**"Error, gpa must be in the format X.X"**);  
 exit(1);  
 }  
  
 printf(**"Please input the student's id number:\n"**);  
 fgets(student\_id, **sizeof** student\_id, **stdin**);  
 fflush(**stdin**);  
 **if**(strlen(student\_id) < 0 || strlen(student\_id) > 6)  
 {  
 printf(**"Error, student id should be format XXXXX."**);  
 exit(1);  
 }  
  
 strcpy(s.name, name);  
 strcpy(s.class, class);  
 strcpy(s.gpa, gpa);  
 strcpy(s.student\_id, student\_id);  
  
 **return** s;  
}

# Part1 Stack Overflow Analysis

## Diagram

24 Bytes to reach.  
32 Bytes to overflow

|  |  |  |
| --- | --- | --- |
| **Element** | **Address** | **Size in Bytes** |
| EIP (return address) | 0xbffff00c | 4 |
| EBP (points to top of stack) | 0xbffff008 | 4 |
| … | … |  |
| class | 0xbfffeff0 | 12 |
| name (**vulnerable**) | 0xbfffefe4 | 12 |
| student\_id | 0xbfffefdc | 8 |
| gpa | 0xbfffefd8 | 4 |

## Program 2: Heap Overflow

### Main\_Program.c

#include **<stdlib.h>**#include **<stdio.h>**#include **"student.h"**#include **"Part1.h"**#include **"Part2.h"**Student\_t s;  
  
**int** main(**int** argc, **const char**\* argv[])  
{  
 printf(**"\nStudent Creation Begin\n"**);  
 *//s = create\_student\_bo();* s = create\_student\_ho();  
 student\_to\_string(s);  
}

### Part2.c

#include **<stdlib.h>**#include **<stdio.h>**#include **<memory.h>**#include **"Part2.h"**#include **"student.h"**Student\_t s;  
  
**struct** Student create\_student\_ho(){  
 **char** \* name, \* class, \* gpa, \* student\_id;  
  
 printf(**"\nPlease print student's name:\n"**);  
 name = malloc(**sizeof**(**char** \*) \* 12);  
 gets(name);  
  
 printf(**"\nPlease print student's class\n"**);  
 class = malloc(**sizeof**(**char** \*) \* 12);  
 fgets(class, **sizeof** class, **stdin**);  
  
 printf(**"\nPlease print the student's gpa:\n"**);  
 gpa = malloc(**sizeof**(**char** \*) \* 4);  
 fgets(class, **sizeof** gpa, **stdin**);  
  
 printf(**"\nPlease print the student's id:\n"**);  
 student\_id = malloc(**sizeof**(**char** \*) \* 8);  
 fgets(student\_id, **sizeof** student\_id, **stdin**);  
  
 strcpy(s.name, name);  
 strcpy(s.class, class);  
 strcpy(s.gpa, gpa);  
 strcpy(s.student\_id, student\_id);  
  
 **return** s;  
};

# Part 2 Heap Overflow Analysis

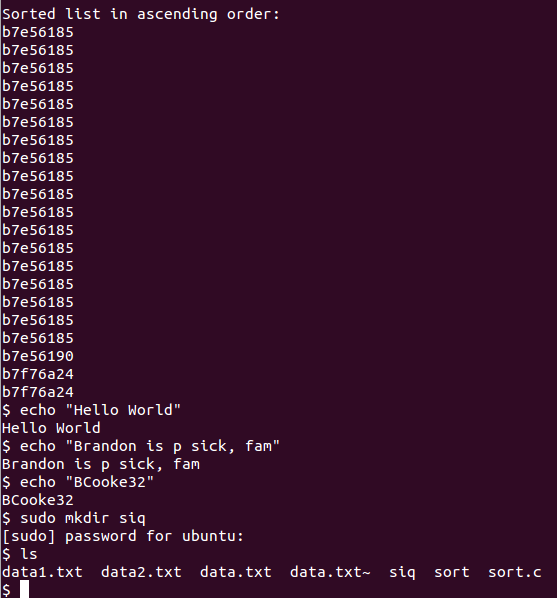
## Diagram

0xbffff00c -0x804b008 =  
3086696452   
Bytes to overflow

|  |  |  |
| --- | --- | --- |
| **Element** | **Address/Content** | **Size in Bytes** |
| EIP (return address) | 0xbffff00c | 4 |
| EBP (points to top of stack) | 0xbffff008 | 4 |
| … | … |  |
| metadata for  student\_id |  |  |
| student\_id (didn’t take input) | 0x804b090 **“an\n”** (**3 bytes**) | malloc(sizeof(char\*) \* 8) |
| metadata for  gpa |  |  |
| gpa (**overwritten**) (didn’t take input) | 0x804b078 **“w” \* 20** (**20 bytes**) | malloc(sizeof(char\*) \* 4) |
| metadata for  class |  |  |
| class input: “Freshman” | 0x804b040 **“shm”** (**3 bytes**) | malloc(sizeof(char\*) \* 12) |
| metadata for name |  |  |
| name (**vulnerable**) input: 184 “w” ’s | 0x804b008 **“w” \* 52** (**52 bytes**) | malloc(sizeof(char\*) \* 12) |

# Exploiting Buffer Overflow

## Data.txt Screenshot

b7e56185

b7e56185

b7e56185

b7e56185

b7e56185

b7e56185

b7e56185

b7e56185

b7e56185

b7e56185

b7e56185

b7e56185

b7e56185

b7e56185

b7e56185

b7e56185

b7e56185

b7e56185

b7e56190

b7f76a24

b7f76a24

# Open Question