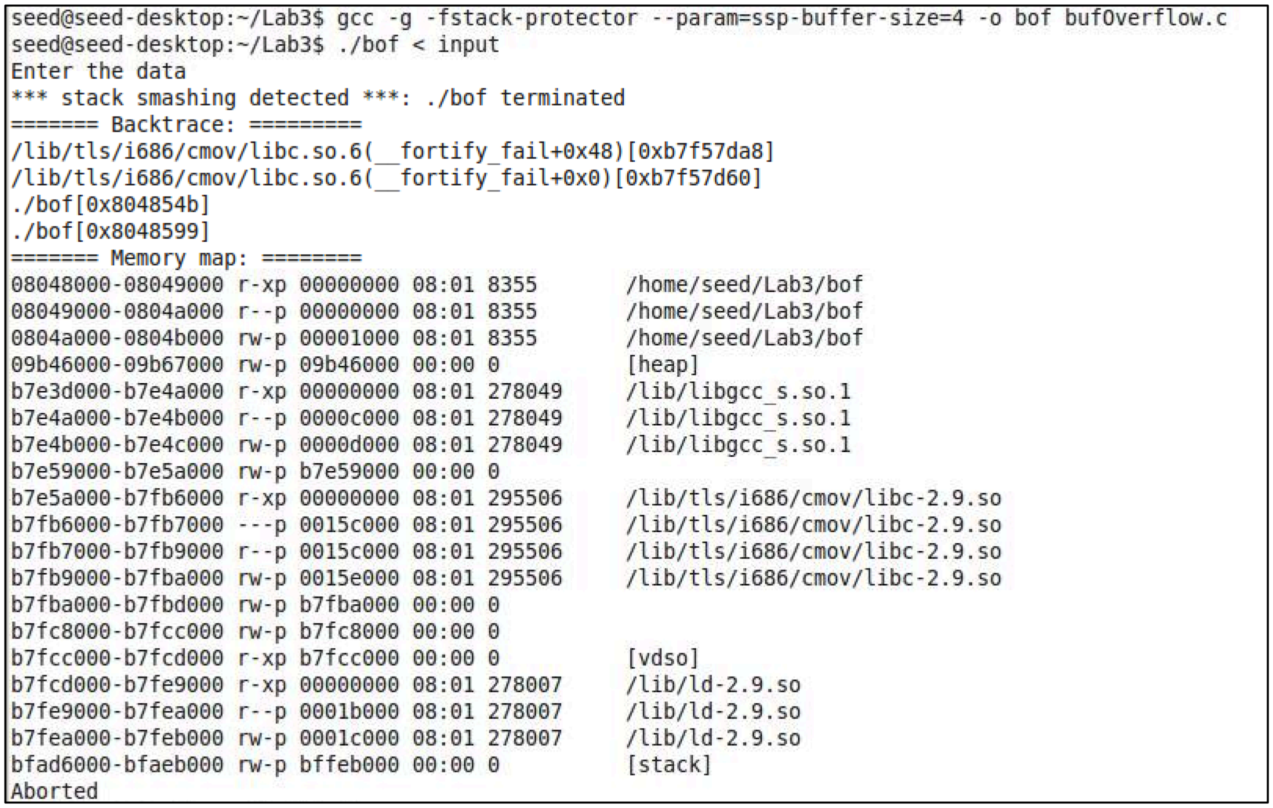
NAME : ***LAKSHITHA RAJ VASANADU***
ID : ***00001115006***

LAB ASSIGNMENT 3 TASK 1
MITIGATION STRATEGIES AND EXPLORING THE STACK

a) Both –fstack-protector and –fstack-protector-all do not work with the given program.

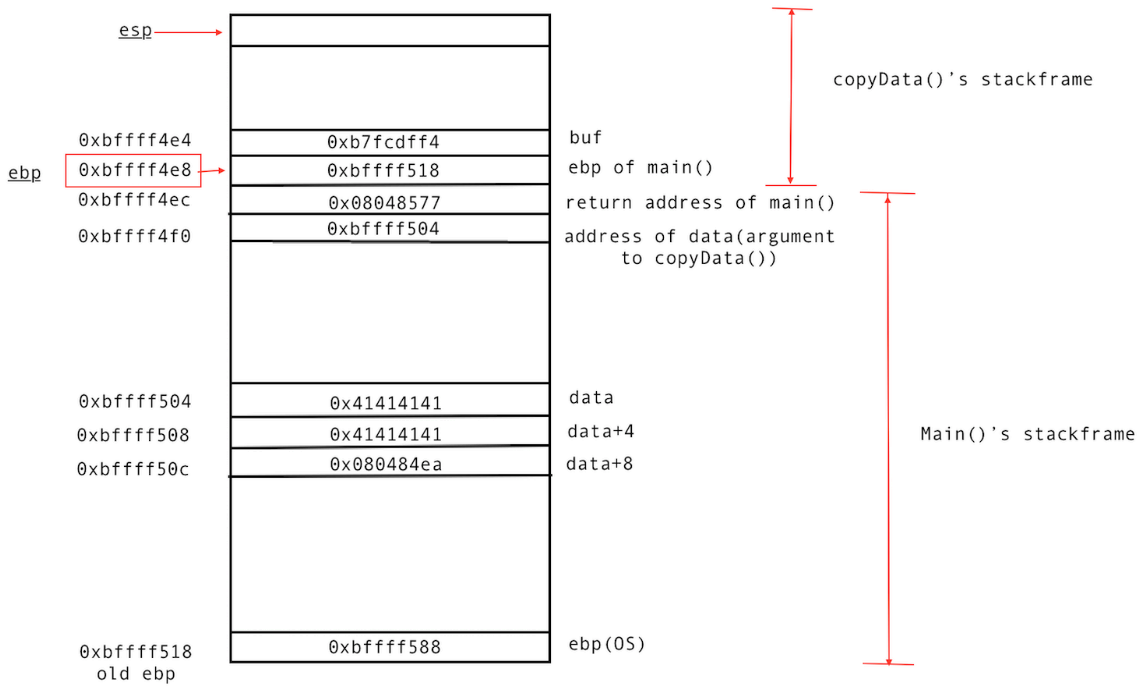
- **-fstack-protector** : This option emits extra code to check for buffer overflows by adding a guard variable to functions with vulnerable objects like *alloca* or buffers with size > 8 bytes.
- **-fstack-protector-all** : This does the same as above but checks for stack smashing in every function.

Both of these work only for buffer > 8 bytes. In the given code, buffer size is 4 (**char buf[4]**) and hence it does not work. Instead, the option **--param=ssp-buffer-size=4** can be used with gcc to explicitly specify the buffer size. This helps to protect the code from buffer overflows.

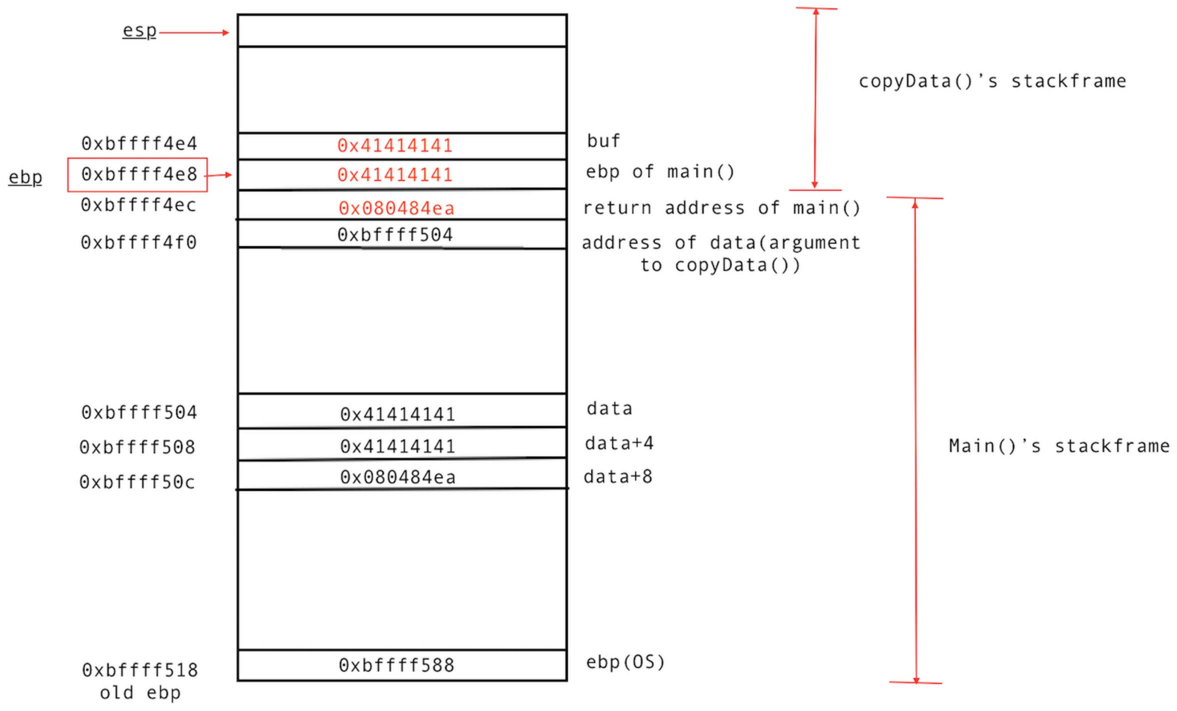


```
seed@seed-desktop:~/Lab3$ gcc -g -fstack-protector --param=ssp-buffer-size=4 -o bof bufOverflow.c
seed@seed-desktop:~/Lab3$ ./bof < input
Enter the data
*** stack smashing detected ***: ./bof terminated
======= Backtrace: =========
/lib/tls/i686/cmov/libc.so.6(__fortify_fail+0x48)[0xb7f57da8]
/lib/tls/i686/cmov/libc.so.6(__fortify_fail+0x0)[0xb7f57d60]
./bof[0x804854b]
./bof[0x8048599]
======= Memory map: ========
08048000-08049000 r-xp 00000000 08:01 8355       /home/seed/Lab3/bof
08049000-0804a000 r--p 00000000 08:01 8355       /home/seed/Lab3/bof
0804a000-0804b000 rw-p 00001000 08:01 8355       /home/seed/Lab3/bof
09b46000-09b67000 rw-p 09b46000 00:00 0          [heap]
b7e3d000-b7e4a000 r-xp 00000000 08:01 278049     /lib/libgcc_s.so.1
b7e4a000-b7e4b000 r--p 0000c000 08:01 278049     /lib/libgcc_s.so.1
b7e4b000-b7e4c000 rw-p 0000d000 08:01 278049     /lib/libgcc_s.so.1
b7e59000-b7e5a000 rw-p b7e59000 00:00 0
b7e5a000-b7fb6000 r-xp 00000000 08:01 295506     /lib/tls/i686/cmov/libc-2.9.so
b7fb6000-b7fb7000 ---p 0015c000 08:01 295506     /lib/tls/i686/cmov/libc-2.9.so
b7fb7000-b7fb9000 r--p 0015c000 08:01 295506     /lib/tls/i686/cmov/libc-2.9.so
b7fb9000-b7fba000 rw-p 0015e000 08:01 295506     /lib/tls/i686/cmov/libc-2.9.so
b7fba000-b7fbd000 rw-p b7fba000 00:00 0
b7fc8000-b7fcc000 rw-p b7fc8000 00:00 0
b7fcc000-b7fcd000 r-xp b7fcc000 00:00 0          [vdso]
b7fcd000-b7fe9000 r-xp 00000000 08:01 278007     /lib/ld-2.9.so
b7fe9000-b7fea000 r--p 0001b000 08:01 278007     /lib/ld-2.9.so
b7fea000-b7feb000 rw-p 0001c000 08:01 278007     /lib/ld-2.9.so
bfad6000-bfaeb000 rw-p bffeb000 00:00 0          [stack]
Aborted
```

b) Before strncpy()



After strncpy()



Here, we can observe that return address in main() is over written by our input.

c) The given program can be re-written as follows to protect from buffer overflows. This code uses **Input Validation** as a mitigation strategy to **prevent** buffer overflow.

Here, we check whether the length of the input parameter to *CopyData* function is less than the buffer size that is statically allocated. If the condition fails, the program in aborted.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void message()  {
  puts(“Should not reach here”);
  exit(0);
}

void copyData(char* str)  {
  char buf[4];

  if(strlen(str) >= sizeof(buf))  {

       printf(“Aborting the program due to long input!!”);
       abort();
  }

  strncpy(buf, str, strlen(str));
}

int main(int argc, char* argv[])  {
  char data[12];
  printf(“Enter the data\n”);
  scanf(“%s”, data);
  copyData(data);

  return 0;
}
```