SANTA CLARA UNIVERSITY
Computer Engineering Department

Lab 3 Task 1

Mitigation Strategies and Exploring the Stack

Total points: 15

Goals

This lab has two objectives:

1. To help you apply one or more of the mitigation strategies discussed in class to prevent buffer overflow.
2. To familiarize you with how the data is stored on the stack. This part of the lab is used to prepare you for the next lab assignment (*return-into-libc*).

Lab Tasks

Before you start work on the following tasks, you must turn off address randomization in the Ubuntu VM. To do this, login as root, and type the following on the command prompt:
$ sysctl –w kernel.randomize_va_space=0

1. (15 points) Examine the program in the folder "Lab 3 task1". This contains the following files:
   bufOverflow.c: a program
   malinput.c: contains C code that is used to generate the contents of file "malinput".

   bufOverflow.c :

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


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

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

malinput.c :

```
#include <stdio.h>

int main(void) {
  FILE *fp;
  fp = fopen("input", "w+");
  fprintf(fp, "%s", "AAAAAAAA\xea\x84\x04\x08");
  fclose(fp);
  return 0;
}
```

Compile and run this program as follows:

```
seed@seed-desktop:~/winter2015/lab3$ gcc -o malinput malinput.c
seed@seed-desktop:~/winter2015/lab3$ ./malinput
seed@seed-desktop:~/winter2015/lab3$ gcc -g -o bof bufOverflow.c
seed@seed-desktop:~/winter2015/lab3$ ./bof < input
Enter the data
Should not reach here
```

Observe that the program prints out "Should not reach here" even though the function message() is never called in the program. Your task is to figure out how this portion of the code gets executed.

(a) Compile and run the program using the gcc options –fstack-protector and –fstack-protector-all. These options are used to detect buffer overflows when the program is being run. Explain why these options may not work with the program bufOverflow.c and how they can be enabled.
(Hint: explore the use of the GCC option --param=ssp-buffer-size=)

(b) Draw a diagram of the stack showing how it looks
   a. immediately *before* the `strncpy()` function is executed. You must show where the argument str, saved base pointer of main, return address of main() and local variable buf are placed on the stack.
   b. immediately *after* the `strncpy()` function completes. *S*how the contents of the buffer and how the input to the program is stored on the stack.

Use gdb to help you complete this task. You must show *stack addresses* and the corresponding *data* at those addresses in your diagram.

(c) Rewrite the program using any one of the mitigation strategies that can be used to *prevent* buffer overflows.