Buffer Overflow Lab

This lab should last for two weeks

Introduction

In this laboratory we will look at the issue of buffer overflow.

Gdb cheat sheet: https://darkdust.net/files/GDB%20Cheat%20Sheet.pdf

Task 1:

In this task, find the ESP, EBP and Return address in the stack frames of the following functions: main, func1, and func2 when the following code is compiled and executed.

```c
#include<stdio.h>

void fun2(int a, int b){

        printf("This is in fun2\n");
        printf("The value of a is %d\n", a);
        printf("The value of b is %d\n", b);

}

void fun1(){
        int n=3;
        int m=4;
        fun2(n,m);
        printf("This is in fun1\n");
}

int main(){
        int a=1;
        int b=2;
        fun1();
        printf("end of demo\n");
        return 0;

}
```

Task 2:

In this task, discover buffer overflow vulnerability and exploit the program of the following code (i.e. print out the following message in the terminal “You have correctly …Well done!”)

```c
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
#include<string.h>
#include<err.h>
int main(int argc, char **argv){

        volatile int modified;
        char buffer[32];

        if(argc<2){
                errx(1, "Please provide some arguments\n");
        }
        modified = 0;

        strcpy(buffer, argv[1]);

        if(modified != 0x51525354){
                printf("Try again, you got 0x%08x\n", modified);

        }else{

                printf("You have correctly got the variable to the right value. Well done!\n");
        }
        return 0;
}
```

Task 3:

In this task, discover buffer overflow vulnerability and exploit the program of the following code (e.g. executing the for loop in the vde_msg() function)

```c
#include<stdio.h>
#include<string.h>
#include<err.h>
#include<unistd.h>
void vde_msg(){
        char i;
        for (i=0; i<128; i++){
                printf("VDE is cooooooooool!\n");
                sleep(5);
        }

}

void getMessage(){
        char buffer[32];
        printf("Enter a message: ");
        gets(buffer);
        printf("You entered: %s\n", buffer);

}
int main(int argc, char **argv){

        printf("In main.\n");
        printf("Calling getMessage.\n");
        getMessage();

        printf("Back in main.\n");

        return 0;
}
```

Additional task

Download the serial.c file from moodle. Then compile it and run the program. Try various inputs and see if you can discover any vulnerability and exploit the program (i.e. print the “The serial number is valid” on the command line).