CS4615 - P3 - BUFFER OVERFLOW PRACTICAL

Overview

In the lectures we have looked at buffer overflows. In this practical session we investigate a simple c program emulating a banking application. The application loads a customer database into memory, holding first and last name and account balance. The application can be used to look at customer data including balance and can be used to modify the name record (change first or last name). However, the application is not designed to change the account balance. For example, the application might be used in a call centre to deal with changes to customer details.

Data Structure

The application populates the data structure *customers* which holds the customer data. This data structure is an array of structs of the type *customer* which has the following format:

```
//customer data structure
struct customer{
        char    f_name[10];
        char    l_name[15];
        int     balance;
};

//customer database as array of structs
struct customer customers[NUM_C];
```

Compiling

Depending on the gcc version and platform you may have to compile the program differently. On the Linux system cs1.ucc.ie use:

```
gcc bank.c -o bank
```

On other platforms you may have to use the compiler flag *FORTIFY_SOURCE* as per default protection is enabled:

```
gcc -O2 -D_FORTIFY_SOURCE=0 bank.c -o bank
```

PART1: Code Analysis

Write a function to print out the binary representation of the structure *customers*. Have a look at how much space is used for *f_name* and *l_name* and balance. Investigate how the data structure is located in memory. Which parts of the code can access the data structure and modify it?

PART2: Balance Manipulation

Simply using the provided interface, used to alter first and last name, try to manipulate a customers balance.

PART3: Balance Manipulation - Maximum Balance?

Try to change a balance to the maximum amount possible. What is the maximum amount that the balance can be set to?

PART4: FORTIFY_SOURCE

Now compile the program with option *-D_FORTIFY_SOURCE=2*. What is the effect? Investigate what the compile flag *FORTIFY_SOURCE* does.

```
gcc -O2 -D_FORTIFY_SOURCE=2 bank.c -o bank
```

PART5: Program Fixing

Fix the program by using safe string operations. How does this approach compare to the *FORTIFY_SOURCE* compiler option? Are both approaches equivalent?

PART6: Fixing Data Structures

Assume we could not change the program (e.g. the used functions are in a library we cannot modify). Could the issue be avoided by simply using a different data structure?

CS4615 Continuous Assessment - PART 3

Please submit an answer to the following question with your CS4615 Continuous Assessment. Your answer should not be longer than half a page (You can use figures or code pieces to illustrate your answer).

Question P3 [2 MARKS]: Balance Manipulation - Maximum Balance?

What is the maximum amount that the balance can be set to using a buffer overflow if you can only use lower and upper case letters? Explain in detail how you arrived at your answer.