中国科学技术大学 软件学院	***University of Science and Technology of China*** ***School of Software Engineering of USTC***

Lab2: Defusing a Binary Bomb

G430113385: Computer Systems A Programmer’s Perspectives

2015-12-09

Junmin Wu (jmwu@ustc.edu.cn) is the lead person for this assignment.

TA: Xiaodong Zhu(xdzhu001@mail.ustc.edu.cn)

1 Introduction

A binary bomb is a program that consists of a sequence of phases. Each phase expects you to type a particular string on *stdin*. If you type the correct string, then the phase is *defused* and the bomb proceeds to the next phase. Otherwise, the bomb *explodes* by printing "BOOM!!!" and then terminating. The bomb is defused when every phase has been defused.
There are too many bombs for us to deal with, so we are giving each group a bomb to defuse. Your mission, which you have no choice but to accept, is to defuse your bomb before the due date. Good luck, and welcome to the bomb squad!

2 Step 1: Get Your Bomb

Each group of students will attempt to defuse their own personalized bomb. Each bomb is a Linux binary executable file that has been compiled from a C program. To obtain your group's bomb, you can login in the server (sse.ustc.edu.cn) and download the tar file named ID1+ID2 where ID1 is the Student Number of the first team member and ID2 is the Student Number of the second team member.
Then give the command: tar xvf ID1+ID2.tar. This will create a directory called ./bomb with the following files:

- README: Identifies the bomb and its owners.
- bomb: The executable binary bomb.
- bomb.c: Source file with the bomb's main routine.

3 Step 2: Defuse Your Bomb

Your job is to defuse the bomb.You can use many tools to help you with this; please look at the hints section for some tips and ideas. The best way is to use your favorite debugger to step through the disassembled binary.
Each phase is worth 10 points, for a total of 60 points.The phases get progressively harder to defuse, but the expertise you gain as you move from phase to phase should offset this difficulty. However, the last phase will challenge even the best students, so please don't wait until the last minute to start.
The bomb ignores blank input lines. If you run your bomb with a command line argument, for example,

```
linux> ./bomb solution.txt
```

then it will read the input lines from solution.txt until it reaches EOF (end of file), and then switch over to stdin.
To avoid accidently detonating the bomb, you will need to learn how to single-step through the assembly code and how to set breakpoints. You will also need to learn how to inspect both the registers and the memory states. One of the nice side-effects of doing the lab is that you will get very good at using a debugger. This is a crucial skill that will pay big dividends the rest of your career.

4 Hints *(Please read this!)*

There are many ways of defusing your bomb. You can examine it in great detail without ever running the program, and figure out exactly what it does. This is a useful technique, but it not always easy to do. You can also run it under a debugger, watch what it does step by step, and use this information to defuse it. This is probably the fastest way of defusing it.
There are many tools which are designed to help you figure out both how programs work, and what is wrong when they don't work. Here is a list of some of the tools you may find useful in analyzing your bomb, and hints on how to use them.

- gdb
  - To keep the bomb from blowing up every time you type in a wrong input, you'll want to learn how to set breakpoints.
  - The CS:APP Student Site at http://csapp.cs.cmu.edu/public/students.html has a very handy single-page gdb summary.
  - For other documentation, type "help" at the gdb command prompt, or type "man gdb", or "info gdb" at a Unix prompt. Some people also like to run gdb under gdb-mode in emacs.
- objdump –t

This will print out the bomb's symbol table. The symbol table includes the names of all functions and global variables in the bomb, the names of all the functions the bomb calls, and their addresses. You may learn something by looking at the function names!

- objdump –d

Use this to disassemble all of the code in the bomb. You can also just look at individual functions.Reading the assembler code can tell you how the bomb works.
Although objdump -d gives you a lot of information, it doesn't tell you the whole story. Calls to system-level functions are displayed in a cryptic form. For example, a call to sscanf might appear as:

```
8048c36:   e8    99    fc    ff    ff       call 80488d4 <_init+0x1a0>
```

To determine that the call was to sscanf, you would need to disassemble within gdb.

- Strings

This utility will display the printable strings in your bomb.

5 Hand In Instructions

- Send your solution.txt to the TA (sse.ustc.edu.cn), the attachment is your own solution.txt, and the subject of your mail is as follows:
  - "Lab2 [ID]" where ID is your Student Number, if you are working alone, or
  - "Lab2 [ID1+ID2]" where ID1 is the Student Number of the first team member and ID2 is the Student Number of the second team member.
- Please check out the list of the teams who have turned in the codes on the ftp server.
- If you have turned in the codes, but your name doesn't appear in the list, please contact to the TA.