CS 211 - Computer Architecture
Instructor: Prof. Santosh Nagarakatte

Programming Assignment 03
Disassembling and Defusing a Binary Bomb

March 3, 2017
Due by March 27 - 5:00PM

dis·as·sem·ble (vt)
take apart: to take something such as a piece of machinery apart.
– Bing Dictionary

0 Overview

The purpose of Programming Assignment 3 (PA3) is for you to become familiar with **x86-IA32** Instruction Set Architecture (ISA).

The nefarious Dr. Evil has planted a slew of "binary bombs" on our machines. 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 everyone 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!

1 Instructions

The bombs were constructed specifically for **32-bit machines** and **Linux** operating system. You **must** do this assignment on the **iLab machines**. You will not defuse the bomb otherwise and will not get credit.

In fact, there is a rumor that Dr. Evil has ensured the bomb will always blow up if run elsewhere. There are several other tamper-proofing devices built into the bomb as well, or so they say.

Open an Internet Browser and go to the URL: `http://airavat.cs.rutgers.edu:17200`. This address is only "visible" when you are connected in the Rutgers Network. Fill the form up with your NetID and your email address to get your bomb package. The file that you will get is in the format bombN.tar, where N is your bomb ID, i.e. YOUR ID.

If you haven't downloaded it in the iLab machines, copy the file to there and untar your bomb into your home directory.

We recommend that you download no more than two bombs. Copy the bomb that you downloaded into your iLab account and work with it.

```
$ tar -xvf bomb<ID>.tar
```

It will create a directory **bomb**$< ID >$ that should contain the following files:

- **bomb**: The executable binary bomb
- **bomb.c**: Source file with the bomb's main routine
- **README**: File with the bomb ID and extra information

Your job is to defuse the bomb. You can use many tools to help you with this; please look at the tools section for some tips and ideas. The best way is to use a debugger to step through the disassembled binary.

The bomb has **9** phases. The phases get progressively harder to defuse, but the expertise you gain as you move from phase to phase should offset this difficulty. Nonetheless, the latter phases are not easy, so please don't wait until the last minute to start. (If you're stumped, check the hints section at the end of this document.)

The bomb ignores blank input lines. If you run your bomb with a command line argument, for example,

```
./bomb defuser.txt
```

then it will read the input lines from **defuser.txt** until it reaches **EOF** (end of file), and then **switch over to stdin** (standard input from the terminal). In a moment of weakness, Dr. Evil added this feature so you don't have to keep retyping the solutions to phases you have already defused.

To avoid accidentally 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.

IMPORTANT: Every time that the bomb explodes, you will lose 0.5 points. It is important that you use breakpoints and avoid those unnecessary explosions.

2 Resources

There are a number of online resources that will help you understand any assembly instructions you may encounter while examining the bomb. In particular, the programming manuals for **x86-IA32** processors distributed by Intel and AMD are exceptionally valuable. They both describe the same ISA, but sometimes one may be easier to understand than the other.

2.1 Useful for this Lab

- Intel Instruction Reference

It is important to realize that the assembly syntax of the instructions on the Intel manual follows the Intel assembly language, while in the book, in gcc, and in gdb they all use the AT&T assembly language. They are perfectly interchangable, you can identify the differences in this webpage:
http://asm.sourceforge.net/articles/linasm.html

2.2 Not Directly Useful, but Good Brainfood Nonetheless

- Intel 64 and IA-32 Architectures Software Developer's Manual Volume 1: Basic Architecture
- Intel 64 and IA-32 Architectures Software Developer's Manual Volume 3: System Programming Guide

2.3 Checking your Work

We provided a webpage where you can check your work. Access:
http://airavat.cs.rutgers.edu:17200/scoreboard to verify how many points you have, up to which phase you have defused the bomb, and so on. You have to be on the Rutgers network to access the above link.

3 Tools

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.

We do make one request, please do not use brute force! You could write a program that will try every possible key to find the right one, but the number of possibilities is so large that you won't be able to try them all in time.

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**: The GNU debugger is a command line debugger tool available on virtually every platform. You can trace through a program line by line, examine memory and registers, look at both the source code and assembly code (we are not giving you the source code for most of your bomb), set breakpoints, set memory watch points, and write scripts. Here are some tips for using 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 has a very handy **gdb summary** (there is also a more extensive tutorial).
  - 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 bomb**: 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 bomb**: 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 may look cryptic. 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 -t x bomb**: This utility will display the printable strings in your bomb and their offset within the bomb.

Looking for a particular tool? How about documentation? Don't forget, the commands **apropos** and **man** are your friends. In particular, **man ascii** is more useful than you'd think. If you get stumped, use the course's discussion board on sakai.

4 Submission

You have to e-submit the assignment using Sakai. Your submission should be a tar file named $bomb < YOURID > .tar$ that can be extracted using the command:

```
tar -xf bomb<ID>.tar
```

Extracting your tar file must give a directory called $bomb < ID >$. This directory should contain the same files that you downloaded, along with the file defuser.txt to defuse the bomb.

To create the tar file that you will submit after finishing your programming assignment, you will use the following command line, in the parent directory of $bomb < ID >$:

```
tar -cvf bomb<ID>.tar bomb<ID>/
```

5 Grading

Your grade will be based on how many stages of the bomb you have defused. Be careful to follow all instructions. If something doesn't seem right, ask.

6 Collaboration

You are not supposed to assist your friends or other students in solving the bombs. You are required not to use any online resource other than ones explicitly described above. If in doubt, ask. Keep in mind that your final and midterm will test you based on the skills learned in this assignment.