A1: Bomblab

Start Assignment

Due Apr 24 by 11:59pm **Points** 100 **Submitting** a file upload
Available Apr 6 at 11:55pm - Apr 25 at 6am

A1: Bomblab

0. Abstract

Understanding how assembly works is tantamount to understanding how your source code works. Nearly anyone can drive a car, but if you are going to get the most performance out of a vehicle, you need to understand what it is capable of, how it works and just how far you can push it in various circumstances. Stunt drivers, race drivers, commerical drivers and the like all need to understand more about their machinery than just how to operate it. If they understand what happens when they use the controls they can make better use of the controls. Likewise, your job as a computer scientist is not to just write code, nearly anyone can do that, but to be a stunt coder.

The purpose of this assignment is for you to become familiar with the x86 Instruction Set Architecture (ISA). The nefarious Dr. F. Ocsicnar 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 is to defuse your bomb before the due date. Good luck, and welcome to the bomb squad!

1. Introduction

To download your bomb, go to:

http://kill.cs.rutgers.edu:17200/

.. and fill out the form with your NetID and your email address. Your bomb package will be emailed to you. The file that you will get wil be named bombN.tar, where N is your bomb ID. Transfer the bomb package in to your home directory on an iLab machine. This assignment will only work on the iLab machines.

You can then untar the bomb with:

tar -xvf bombID.tar

This will create the directory bombID that should contain the following files:

bomb: The executable binary bomb

bomb.c: (partial) source file with the bomb’s main routine

Be careful! The bomb is shipped to you live! Every time the bomb is invoked and runs until it encounters an incorrect phase code, it will explode.

IMPORTANT: Every explosion will cost you 2 points!

NOTE: You start with your bomb in a digital "bunker", so you can tolerate three explosions without point loss.

.. i.e. You start with 6 bonus points. If you can defuse the bomb with 0 explosions, you keep them.

If you start the bomb, you'll want to Ctrl+C to quit it if you do not know the phase code(s).

If you load the bomb in GDB, be sure to set breakpoints.

2. Methodology

Your job is to defuse the bomb. The bomb has multiple phases and each requires some type of code to be entered in order to defuse it. 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.

Phases must always be answered in order. Once you determine the code to enter for a particular phase, it will not change from run to run.

You can partially automate the code entry for phases you've already cracked by running the bomb with an input file:

./bomb mysolution.txt

.. It will expect one code per line. When it hits the end of the file, it will expect subsequent phase codes to come from stdin.

To avoid accidentally detonating the bomb, you will need to single-step through the assembly code and set breakpoints. You will also need to inspect both the registers and the memory states to determine the answers for the phases. 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.

We provided a webpage where you can check your work. Here you can access the scoreboard to verify how many points you have, up to which phase you have defused the bomb, and so on.

http://kill.cs.rutgers.edu:17200/scoreboard

3. Results

There are many methods of defusing your bomb. You should use GDB to good effect.

We strongly recommend you do not attempt to brute force the solutions. Every incorrect answer can result in an explosion, which will cost you points. Brute forcing an answer would be extremely

inefficient and likely quickly result in a 0. Likely the fastest way to defuse it is to run it in GDB, watch what it does step by step, and use this information to defuse it.

Some tools you may want to use are:

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, youll want to learn how to set breakpoints.

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.

The recitation slides have a very handy gdb summary

objdump -t bomb: This will print out the bombs 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.

strings -t x bomb: This utility will display the printable strings in your bomb and their offset within the bomb. Don't forget, the commands apropos and man are your friends.

4. Submission

In order to submit your work on Canvas you should first add a solution file named "mysolution.txt" to your bomb directory. That file should hold the solutions to all phases of your bomb, one per line.

Before submitting, make sure that if you run:

```
./bomb mysolution.txt
```

.. that the bomb is fully defused.

Make sure that your bomb and bomb.c are still in the bomb directory before repacking it.

Then, repack your bomb directory in to a tar using:

```
tar -cvf bombID.tar bombID
```

.. this will result in a file named bombID.tar

Submit that file via Canvas.