CS 211: Computer Architecture, Fall 2017
Programming Assignment 2: Art of the Bomb

In order to maintain the apprehension and disquiet of Halloween Dr. Evil has distributed digital bombs across the iLabs. Thankfully, Evil's Doctorate is in the Fine and Performing Arts, not Computer Science, so you are entirely capable of defusing any of the bombs by examining its assembly using GDB. There are too many bombs for one person to defuse, so you must each download one of Evil's bombs and defuse it. Each bomb consists of 9 phases. You must enter the code sequence on STDIN in order to defuse each phase. If you enter the correct code string for a phase the bomb will advance to the next phase. If you enter an incorrect code, the bomb will explode. It will print out "BOOM!!!" and terminate. The bomb will be completely defused if you enter the code sequence for all nine phases. If you want to quit where you are be sure to break out of the bomb. Exploding it by giving it bad code sequences will cost you points.

0. 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://ls.cs.rutgers.edu:17660. This address is only "visible" when you are connected in the Rutgers network. Fill out the form 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. If you haven't downloaded it on to the iLab machines, copy the file to there and untar your bomb into your home directory.

$ tar -xvf bomb<ID>.tar

... 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

You can use many tools to help you defuse the bomb. 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 ignores blank input lines. Instead of inputting the answers to completed phases over and over you can run the bomb with a file that contains the code sequences for all the phases you've solved so far:

./bomb mysolution.txt

The bomb will switch over to STDIN once all the code sequences from the file have been read in. List them in order, one per line.

1. 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.

1.0. Directly useful for this assignment

• Intel Instruction Reference:
http://download.intel.com/products/processor/manual/325383.pdf

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

1.1. Indirectly useful for this assignment

• Intel 64 and IA-32 Architectures Software Developer's Manual Volume 1: Basic Architecture
http://download.intel.com/products/processor/manual/253665.pdf

• Intel 64 and IA-32 Architectures Software Developer's Manual Volume 3: System Programming Guide
http://download.intel.com/products/processor/manual/325384.pdf

1.2. Checking your work

We provided a webpage where you can check your work:

http://ls.cs.rutgers.edu:17660/scoreboard

Look up your bomb's ID to see how many points you have, up to which phase of the bomb you have defused, and so on. You have to be on the Rutgers network to access the above link.

2. Tools

There are many ways of defusing your bomb. You can examine it in great detail without ever running the program. 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, or before you run out of points. Exploding the bomb will cost you 0.5 points.

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 flavor of Linux. 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 an incorrect code sequence, 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): http://csapp.cs.cmu.edu/public/students.html

– 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 f c f f f f 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.

3. Submission

Your submission should be a tar file named bomb<ID> .tar that contains a directory called bomb<ID>. This directory should contain the same files that you downloaded, along with the file mysolution.txt that holds the code sequences for each phase of the bomb so that running ./bomb<ID> mysolution.txt would defuse it.

To create the tar file that you will submit after finishing your programming assignment, run:

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

.. in the parent directory of bomb<ID>

4. Grading

Your grade will be based on how many stages of the bomb you have defused. Be careful to follow all instructions. Exploding the bomb will cost you half a point.

If you have really, really, really screwed up you *can* download a second bomb, however you will have to start your defusing over from scratch as each bomb is distinct.

No one is allowed more than two bombs. Do not download a third.

5. Collaboration

You are not supposed to assist your friends or other students in solving the bombs. Keep in mind that your final and midterm will test you based on the skills learned in this assignment. If you find some way to get the answer here without actually figuring the assembly out, you risk doing much worse on the exams, which are worth much more than this one assignment.