CENG709 Fall 2018
Assignment 1: Defusing a Binary Bomb
Assigned: Oct. 30, 2018 Tuesday

Due: Nov. 20, 2018 Tuesday at 23.55

Please read carefully.

1 Introduction

The nefarious *Dr. Evil* has planted a slew of "binary bombs" on our class 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 each student 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!

Step 1: Get Your Bomb

You can obtain your bomb by pointing your Web browser at:

```
http://144.122.225.203:15213/
```

This will display a binary bomb request form for you to fill in. Enter your user name and email address and hit the Submit button. The server will build your bomb and return it to your browser in a `tar` file called `bombk.tar`, where *k* is the unique number of your bomb.

Save the `bombk.tar` file to a (protected) directory in which you plan to do your work. Then give the command: `tar -xvf bombk.tar`. This will create a directory called `./bombk` 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 and a friendly greeting from Dr. Evil.
- `writeup.{pdf,ps}`: The lab writeup.

If for some reason you request multiple bombs, this is not a problem. Choose one bomb to work on and delete the rest.

Step 2: Defuse Your Bomb

Your job for this lab is to defuse your bomb.

You must do the assignment on one of the class machines. In fact, there is a rumor that Dr. Evil really is evil, and the bomb will always blow up if run elsewhere. There are several other tamper-proofing devices built into the bomb as well, or so we hear.

You can use many tools to help you defuse your bomb. 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 time your bomb explodes it notifies the bomblab server, and you lose 1/2 point (up to a max of 20 points) in the final score for the lab. So there are consequences to exploding the bomb. You must be careful!

The first four phases are worth 10 points each. Phases 5 and 6 are a little more difficult, so they are worth 15 points each. So the maximum score you can get is 70 points.

Although phases get progressively harder to defuse, 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 psol.txt
```

then it will read the input lines from `psol.txt` until it reaches EOF (end of file), and then switch over to `stdin`. 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.

Logistics

This is an individual assignment. All handins are electronic. Clarifications and corrections will be posted on Piazza.

Handin

There is no explicit handin. The bomb will notify your instructor automatically about your progress as you work on it. You can keep track of how you are doing by looking at the class scoreboard at:

```
http://144.122.225.203:15213/scoreboard
```

This web page is updated continuously to show the progress for each bomb.

How to Obtain/Work on Your Bomb

The important part is how you will accomplish your tasks in terms of the environment where you will be solving your bomb. The instructor will have access to your scores through a computer which will act like a server for your bomb notifications to update your status after whether you defused a phase or led it to an explosion. The server has a table of hostnames which are considered valid hostnames in the network. The valid hostnames are determined to be inek machines located in the computer labs of METU Computer Engineering Department (namely MERA, DIGITAL and ISMAILABI). What this means is you will solve your bombs in the inek machines. Even if you tried to solve it in your personal desktops/laptops, since their hostnames will not be listed in the server's table, you will get an error while using `./bomb<number>` in your personal computer. Although this restricts your working environment, you do not have to be in front of one of the ineks all the time during your assignment. Once you download your bomb to an inek machine, you can connect via `ssh` to the ineks and do your homework from your personal computer. However, to download your bomb, you have to enter some information of yours to the related boxes in the form when you connect to the server with the link given above. At that time, you have to be in front of an inek machine to fill your credentials easily.

Connecting to ineks via ssh can be done in the following way:

- `ssh eXXXXXXX@external.ceng.metu.edu.tr -p 8085`

  XXXXXXX is where your student id should be with all 7 digits and it should be recognized by the Computer Engineering Department (i.e. you have a valid departmental account, you can use *Ceng On the Web - CoW*), if not, you may need to contact with Student Affairs at the department. If this command gives an error, you can use the command below instead:

  `ssh eXXXXXXX@divan.ceng.metu.edu.tr -p 8085`

- Your CENG password will be prompted. To further your access you have to enter that password and if permission is granted, you will have to use ssh again to connect one of the ineks (you can check the status of inek machines from http://ceng.metu.edu.tr/ineks.html , some of them has an `UNREACHABLE` tag meaning that those machines will not be a help to you at all). You will use the following:

  `ssh inekXX.ceng.metu.edu.tr`

- XX is where the number of the inek machine of interest should be (e.g. inek6, inek34, inek100). Your CENG password will be prompted, again. After entering it and getting the permission to access, you will be connected to an inek and can work on solving your bomb.

Looking for a particular tool? How about documentation? Don't forget, the commands `apropos`, `man`, and `info` are your friends. In particular, `man ascii` might come in useful. `info gas` will give you more than you ever wanted to know about the GNU Assembler. Also, the web may also be a treasure trove of information.

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.

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 this is no good for several reasons:

- You lose 1/2 point (up to a max of 20 points) every time you guess incorrectly and the bomb explodes.
- Every time you guess wrong, a message is sent to the bomblab server. You could very quickly saturate the network with these messages, and cause the system administrators to revoke your computer access.
- We haven't told you how long the strings are, nor have we told you what characters are in them. Even if you made the (incorrect) assumptions that they all are less than 80 characters long and only contain letters, then you will have 26^{80} guesses for each phase. This will take a very long time to run, and you will not get the answer before the assignment is due.

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, this 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.

  The CS:APP web site

  ```
  http://csapp.cs.cmu.edu/public/students.html
  ```

  has a very handy single-page `gdb` summary that you can print out and use as a reference. Here are some other 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.

  - For online 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.

Regulations and Grading

- **Discussion:** Please refer to **Piazza** for questions and updates related to the assignment.
- The server will be closed at 23:55 on Nov. 20 Tuesday and any further updates to the scores after that point will be dropped and not be evaluated.
- **Grading:** This assignment will be a self-grading process, i.e. when you defuse a phase or explode your bomb, a notification will be sent to the server to update your score.
- **Demo:** After the assignment process, you will demonstrate how you defuse a phase or two of your bomb. The time and location for this session will be announced later, either in a lecture or through Piazza, so stay sharp about this matter.
- Good luck and have fun!