Assignment 4: Defusing a Binary Bomb
Fall 2020

Due Date: December 11, 2020
Contact TA: Farzin Negahbani

Dear Bomb Squad!

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. As the Dr. Evil targeted our school, you are only able to work on your bombs while you are connected to the school network. Your mission, which you have no choice but to accept, is to defuse your bomb before the due date. We are eagerly looking at the scoreboard that reflects how you are moving forward with the bombs. In the followings, you can find how to obtain your bomb and check the scoreboard.

Finally and most preciously, we are sharing information that our agents sent to us from Dr. Evil's lab which can be your tools and maps in this dangerous mission.

Good luck, and welcome to the bomb squad!

Step 1: Get Your Bomb

You can access the following URL and by typing your KU ID and email you can obtain your bomb package.

```
1 http://ku-bomblab.eastus.cloudapp.azure.com:15213
```

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

WARNING: Don't forget that you can only work on your bombs on the **Linuxpool** machines.
You can use the following URL to check the progress for each bomb.

```
1 http://ku-bomblab.eastus.cloudapp.azure.com:15213/scoreboard
```

Step 2: Defuse your Bomb

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,

```
$ ./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.

1. Hints

1.1 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 and use gdb. **Here** is a short video that shows how to use gdb and commands to go through a disassembled code. The basics are the same as what you learned during your **gdb** lab. There are treasures commands in the mentioned video, so watch it carefully.

1.2 objdump -t filename

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!

1.3 objdump -d filename

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.

1.4 strings filename

This utility will display the printable strings in your bomb maybe something can be found there.

2. Oral Assessment

Important Note: We plan to ask randomly selected 10% of students to explain their approach verbally after the assignments are graded. And one may lose full credit if he or she fails from this oral part.

3. Late Submission Policy

- You may use up to 5 grace days (in total) over the course of the semester for the assignments. That is you can submit your solutions without any penalty if you have free grace days left.
- Any additional unapproved late submission will be punished (1 day late: 20% off, 2 days late: 40% off) and no submission after 2 days will be accepted.

4. Academic Integrity

All work on assignments must be done individually unless stated otherwise. You are encouraged to discuss with your classmates about the given assignments, but these discussions should be carried out in an abstract way. That is, discussions related to a particular solution to a specific problem (either in actual code or in the pseudocode) will not be tolerated. In short, turning in someone else's work, in whole or in part, as your own will be considered as a violation of academic integrity. Please note that the former condition also holds for the material found on the web as everything on the web has been written by someone else. See Koç University - Student Code of Conduct.

5. Acknowledgement

This assignment is adapted from CMU CS-311 course contents.