CS 270: Systems Programming **Fall 2020**

Project 2

Assigned: Sept. 16, 2020 *Due: Oct. 2, 2020*

The objective of this project is to teach principles of machine-level programs, as well as general debugger and reverse engineering skills.

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!

2 Get Your Bomb

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

`http://birch.netlab.uky.edu:40506/`

This will display a binary bomb request form for you to fill in. Enter your user name (LinkBlue ID) and email address and hit the Submit button. The project 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/copy the `bombk.tar` file to a (protected) directory in which you plan to do your work on your OpenStack VM. 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.

If for some reason you request multiple bombs, this is not a problem. Choose one bomb to work on and delete the rest. [1]

[1] You are at a disadvantage working on multiple bombs at the same time, because you get the credit for the most advanced phase defused in one of the bombs, but get the penalties for the explosions from all bombs.

3 Defuse Your Bomb

Your job for this project is to defuse your bomb.

You must do the assignment on your OpenStack VM. 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 read 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 project server, and you lose 1/2 point (up to a max of 30 points) in the final score for the project. So there are consequences to exploding the bomb. You must be careful!

There are 5 phases and each phase is worth 20 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. (Note: It is `./bomb psol.txt`, not `./bomb < psol.txt`.)

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 project 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.

4 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 is 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 30 points) every time you guess incorrectly and the bomb explodes.
- Every time you guess wrong, a message is sent to the project 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.

  You can reference our class web page

  http://voip.netlab.uky.edu/~fei/teaching/cs270/

  for several documents about using gdb. 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`.
  - Here are some useful gdb commands:
    - `b phase_1`: set a breakpoint at the first instruction of function `phase_1`.
    - `b *0x400f3b`: set a breakpoint at address `0x400f3b`.
    - `nexti` (or `ni`): Run the next assembly language instruction; skip over function calls.
    - `stepi` (or `si`): Run the next assembly language instruction; do not skip over function calls.
    - `finish`: finish all the instructions in the current function.
    - `p /x $rsi`: print the hex value of register `%rsi`.
    - `p /d $rsi`: print it in decimal format.
    - `x /s $rsi`: print the string in main memory pointed to by register `%rsi`.
    - `x /6dw $rsp`: print out six numbers (in decimal) of four bytes (w) in main memory pointed to by register `%rsp`.
    - `x /3xb 0x7fffffffe068`: print out three numbers (in hex) in the memory content at the address with each number occupying one byte (b: 1, h: 2, w: 4, g: 8)

- `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. Run

  ```
  objdump -d bomb > bomb.s
  ```

  will generate the assembly code `bomb.s`. You can just look at individual functions. Reading the assembly code can tell you how the bomb works to a certain degree. Although `objdump -d` gives you a lot of information, it doesn't tell you the whole story. You will need to run `gdb` to help you figure out how it works.

- `strings`

  This utility will display the printable strings in 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. Also, the web may also be a treasure trove of information.

5 Handin

This is an individual project. 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://birch.netlab.uky.edu:40506/scoreboard`

This web page is updated continuously (with a possible lag of 1 minute) to show the progress for each bomb.

As you defuse the bomb, remember the input you typed, probably in the partial solution file `psol.txt`. Add your bomb number and your name at the end of text file and submit it to CSPortal at

`https://www.cs.uky.edu/csportal`.

You may upload the text file as many times as you like. The CSPortal timestamps your submission with the date/time of the last submission.

It takes time to finish a project. So start early and don't wait until the last minute or last day.