KOÇ UNIVERSITY

COMP 201 - Spring 2023
Assignment 3 - Defusing a Binary Bomb

Assigned: 4 May 2023 23:59, Due: 18 May 2023 23:59

Osman Batur İnce (oince22@ku.edu.tr) is the lead person for this assignment.

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 click **here** and type your KUNET ID and email you can obtain your bomb package. Please don't forget that your KUNET ID is your your email without @ku.edu.tr. Note that the webserver is a bit slow, so **please wait for at least 10 seconds** before hitting the submit button again.

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

We don't use GitHub classroom for this assignment and instead you will obtain your bomb from the provided URL. For copying file from your own machine to the LinuxPool accounts you may want to look at scp (copying files and folders over ssh connection) command.

WARNING: Don't forget that you can only work on your bombs on the LinuxPool machines. You can click this **link** to check the progress for each bomb.

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. 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. 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.

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 FILENAME`

  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.

  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 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!

- `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.

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. If you get stumped, feel free to ask your instructor for help.

Submission

We use Blackboard for this assignment ONLY! You are expected to **submit a psol.txt file** in which each line correspond to a different bomb phase. **In addition, you should submit a `report.pdf` file.** **`report.pdf`** should include descriptions about how you approached to each phase (e.g. things that you tried, and things that worked) and successful terminal screenshots for the working approach that you took. The descriptions should only include important details and should not have any redundancy, they should be as concise as possible without omitting critical information. We expect the following structure:

- Phase 1: <your description>
  <your figure(s)>
- Phase 2: <your description>
  <your figure(s)>
- Phase 3: <your description>
  <your figure(s)>
- **...**

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.

How to use linuxpool.ku.edu.tr linux servers [1]

I Connect to KU VPN (If you are connected to the KU network, you can skip this step.)
See for details: https://confluence.ku.edu.tr/kuhelp/ithelp/it-services/network-and-wireless/vpn-access

II Connect to linuxpool.ku.edu.tr server using SSH (Replace USER with your Koç University username):
`$ ssh USER@linuxpool.ku.edu.tr`
(It will ask your password, type your Koç University password.)

III When you are finished with your work, you can disconnect by typing: `$ exit`

Your connection to the server may drop sometimes. In that case, you need to reconnect.

We advice you to watch the following video about the usage of SSH, which is used to connect remote servers, and SCP, which is used to transfer files between remote servers and your local machine:
https://www.youtube.com/watch?v=rm6pewTcSro

[1] For details, please see the guide on linuxpool that we have announced on Blackboard

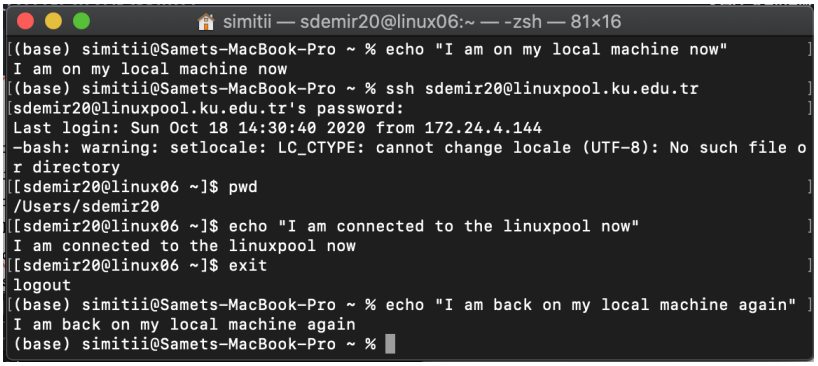

Figure 1: How to connect and disconnect using SSH

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.

Late Submission Policy

You may use up to 7 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.**

Acknowledgement

This assignment is adapted from CMU CS15-213 course contents.