p5: Binary Bombs

Due Apr 21 by 10pm **Points** 90 **Submitting** a file upload
Available until Apr 21 at 10:33pm

This assignment was locked Apr 21 at 10:33pm.

GOALS **BOMBS** **TOOLS** **HINTS** **REQUIREMENTS** **SUBMITTING**

- This project must be done individually.
  - *It is academic misconduct to share your work with others in any form including posting it on publicly accessible web sites, such as GitHub.*
  - *It is academic misconduct for you to copy or use some or all of a program that has been written by someone else.*
- All work for this project is to be done on the **CS Department's instructional Linux machines (https://csl.cs.wisc.edu/services/instructional-facilities)** . You're encouraged to remotely login using the ssh command in the terminal app on Macs, or on Windows using an ssh client such as MobaXterm.
- All projects in this course are graded on **CS Department's instructional Linux machines (https://csl.cs.wisc.edu/services/instructional-facilities)** . To receive credit, make sure your code runs as you expect on these machines.

Learning GOALS

There are two goals for this project. The first is to practice interpreting x86 assembly language. The second is to gain some familiarity with the x86 disassembler, **objdump**, as well as gaining experience using the debugger, **gdb**, with x86.

Binary BOMBS

In this assignment, you will be defusing four "binary bombs". Each bomb is an executable program that prompts the user for five inputs via the stdin console. The user must provide the correct inputs in order to defuse the bomb, but if the wrong input is entered then the bomb explodes! Well, it just prints out the explosion, but you can imagine it having more serious consequences on your grades if you can't figure out the inputs.

Getting the Required Files

The four bombs are unique for every student and are located in the following directory:

```
/p/course/cs354-skrentny/public/students/<your-cs-login-ID>/p5/
```

Replace <your-cs-login-ID> with your actual cs login and copy the contents of the above directory into your private/p5 working directory. There are four executable files named **b1**, **b2**, **b3**, and **b4**. Use your copies to defuse your bombs.

Defusing the Bombs

The challenge is to figure out the correct set of 5 inputs expected by each of the four bombs. When you run your bombs, you'll need to type in your guesses, one at a time, as shown below:

```
[skrentny@jimbo] (55)$ ls
b1* b2* b3* b4*
[skrentny@jimbo] (56)$ ./b1
input 1 (of 5)? 951905
input 2 (of 5)? 1234
BOMB EXPLODED
[skrentny@jimbo] (57)$ ./b1
input 1 (of 5)? 951905
input 2 (of 5)? 994563
input 3 (of 5)? 493693
input 4 (of 5)? 828695
input 5 (of 5)? 278566
success!
[skrentny@jimbo] (58)$
```

Once you've figured out all of the inputs for a bomb, create solution file containing the five lines of input for its associated bomb followed by a single empty line. Name your solution files accordingly: **b1.solution**, **b2.solution**, **b3.solution**, and **b4.solution**. We will test your solution files using input redirection with the binary bomb executables as show below.

```
[skrentny@jimbo] (77)$ ls
b1*  b1.solution  b2*  b2.solution  b3*  b3.solution  b4*  b4.solution
[skrentny@jimbo] (78)$ cat b1.solution
951905
994563
493693
828695
278566
[skrentny@jimbo] (79)$ ./b1 < b1.solution
input 1 (of 5)? input 2 (of 5)? input 3 (of 5)? input 4 (of 5)? input 5 (of 5)? success!
[skrentny@jimbo] (80)$
```

All 5 inputs need to be correct to defuse a bomb. If the bomb explodes, no points will be given no matter how many inputs were correct before the explosion. You can verify your solution files by testing them the same way as shown above.

Make sure you create your solution files with a text editor (vim/gedit/nano) on a CS Department's instructional Linux machine (recall that you can remote access with ssh or MobaXterm). Creating solution files on **Windows or Mac** will result in problems with the end-of-line mark and will cause your solutions to fail when tested with input redirection. Make sure your solution file contains five non-empty lines followed by a single empty line. To create that empty line, **remember to just press enter after the last line in the solution file**. The bomb executable will be trapped into an infinite loop if the solution file contains less than 5 lines. If that happens, press ctrl-c to break out of the program.

TOOLS: Objdump and gdb

To figure out how to defuse your binary bombs, you will use two powerful tools: **objdump** and **gdb**. Both are critical in reverse engineering each binary bomb to determine the inputs.

objdump

objdump is a command in Linux to display information about object files. For this project, the two important command line options are:

- **`-d`** which disassembles a binary
- **`-s`** which displays the full binary contents of the executable

For example, to see the assembly code of bomb b1, you would enter:

```
objdump -d b1
```

Start by looking in **`main()`** to begin figuring out what the code is doing.

The **`-s`** flag is also quite useful as it shows the contents of each segment of the executable. This can be used when looking for the initial value of a given variable.

Use output redirection to save the output of **objdump** in a file, so that you don't have to repeatedly regenerate it every time. Both command line options can be used at the same time to create a full dump of the contents of the executable as well as the disassembled contents.

gdb

By now, you've used the debugger, gdb, to help find segmentation faults, but it's an even more powerful tool in your search for clues to defuse each binary bomb. The following gdb commands in addition to the ones specified in p3 are useful to defuse the bombs:

- **`finish`**: continue until the current function returns and prints the return value (if any)
- **`print`**: prints the contents of variable or memory location or register
- **`set var <variable_name>=<value>`**: changes the content of a variable to the given value

- **`info registers`**: shows you the contents of all of the registers of the system
- **`x/nfu <addr>`**: The examine command, which shows you the contents of memory. **n**, **f**, and **u** are all optional parameters that specify how much memory to display and how to format it. **addr** is the hexadecimal address you want to look at. So for instance "**`x/3ub 0x54320`**" is a request to display 3 bytes (b) of memory formatted as unsigned decimal integers (u) starting at the address 0x54320.

HINTS

- x86 cheat sheet
- Function **`strtol()`** corresponds to the use of **`atoi()`** in C source code.
- Every C program has a **`main()`** function. Figure out how to locate it.
- A loop in **`main()`** iterates five times. Remember that each bomb requires five inputs.
- On a wrong input, function **`bomb()`** is called. This results in an explosion.
- If all five inputs are correct, function **`success()`** is called.
- Function arguments are set up in the call stack just prior to the function call.
- The two parameters of **`strcmp()`** are addresses to two C strings.
- The return value of a function is stored in **%eax**.

REQUIREMENTS

We will test your solution files by running them as shown in the sample output above. It's your responsibility to ensure that your solutions correctly defuse each of the four binary bombs on the CS Linux lab machines.

SUBMITTING Your Work

Submit the following **solution text files** under Project 5 in Assignments on Canvas before the deadline:

1. b1.solution
2. b2.solution
3. b3.solution
4. b4.solution

It is your responsibility to ensure your submission is complete with the correct file names having the correct contents. The following points will seem obvious to most, but we've found we must explicitly state them otherwise some students will request special treatment for their carelessness:

- **You will only receive credit for the files that you submit.** You will not receive credit for files that you do not submit. Forgetting to submit, not submitting all the listed files, or submitting executable files or other wrong files will result in you losing credit for the assignment.
- **Do not zip, compress, submit your files in a folder, or submit each file individually.** Submit only the text files as listed as a single submission.

- **Make sure your file names exactly match those listed.** If you resubmit your work, Canvas will modify the file names by appending a hyphen and a number (e.g., b1-1.solution) and these Canvas modified names are accepted for grading.

Repeated Submission: You may resubmit your work repeatedly until the deadline has passed. **We strongly encourage you to use Canvas as a place to store a back up copy of your work.** If you resubmit, you must resubmit all of your work rather than updating just some of the files.

Project p5

Criteria	Ratings		Pts
Follows the filename and format requirements	**10.0 pts** **Full Marks**	**0.0 pts** **No Marks**	10.0 pts
b1	**20.0 pts** **Full Marks**	**0.0 pts** **No Marks**	20.0 pts
b2	**20.0 pts** **Full Marks**	**0.0 pts** **No Marks**	20.0 pts
b3	**20.0 pts** **Full Marks**	**0.0 pts** **No Marks**	20.0 pts
b4	**20.0 pts** **Full Marks**	**0.0 pts** **No Marks**	20.0 pts
			Total Points: 90.0