HW2: Bomb Lab – Defusing a Binary Bomb

Daeyeon Kim and Yejin Lee
Architecture & Code optimization (ARC) Lab

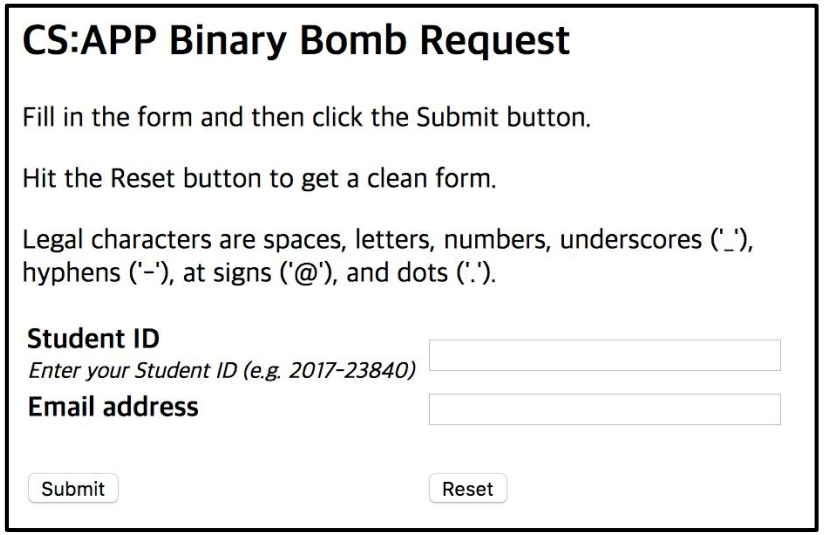
October 2nd, 2018

Introduction

- **The goal of this lab, Defusing a Binary Bomb, is**
  - To defuse a binary bomb using correct strings
- **In this lab,**
  - You will learn the **x86 assembly language**
  - You will learn how to use **a debugger (gdb, objdump)**
- **You must use Martini server (martini.snucse.org)**
  - Can’t run on local machine & other linux environment
  - Students who did not get your martini account should come to TA and receive your ID and PW

Step 1: Get your bomb

- **Access the webpage (http://martini.snucse.org:15213) on internet browser**
  - Your student ID (e.g. 2018-12345) and email address



- **Then you can get "bomb$k$.tar" file (where k is a number)**
  - README: Identifies the bomb and its owners
  - bomb: The executable binary bomb
  - bomb.c: Source file with bomb's main routine

Step 2: Defuse your bomb

- **You need to find correct strings to defuse your bomb**
  - Use a debugger and other tools

```
jane0013@martini:~/bomb1$ ./bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
```

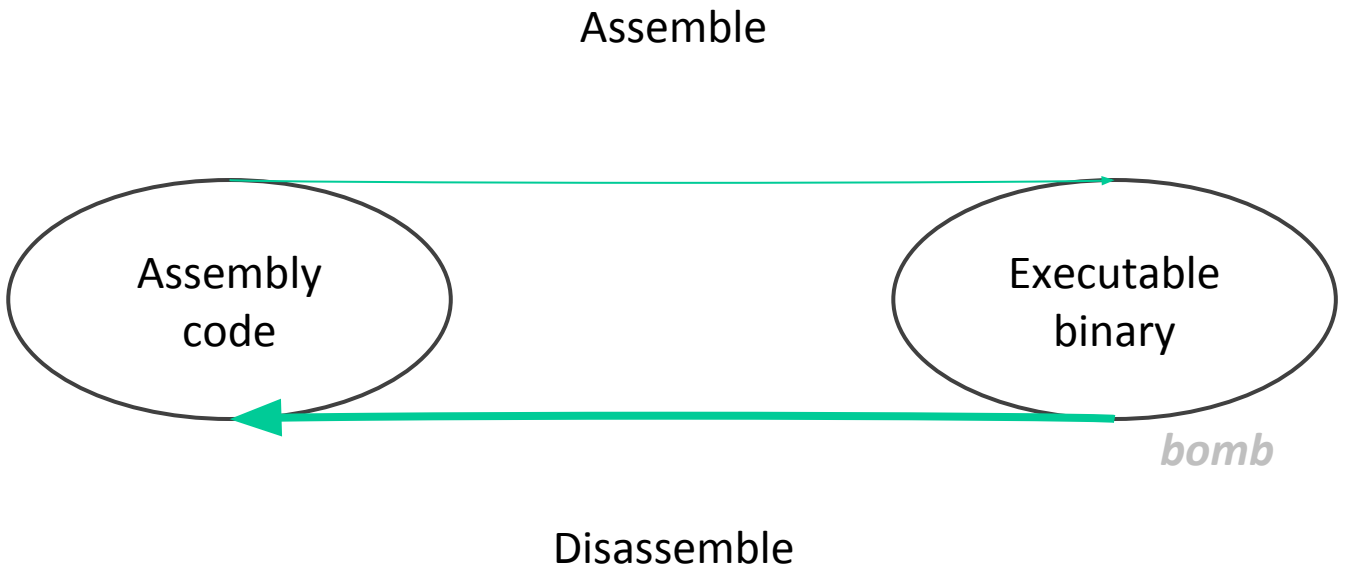
- **Your bomb consists of 6 phases**
  - Each phase has its own correct string
  - If your input does not match, then your bomb will explode

```
Phase 1 defused. How about the next one?
```

```
BOOM!!!
The bomb has blown up.
Your instructor has been notified.
[Inferior 1 (process 29878) exited with code 010]
(gdb)
```

Step 2: Defuse your bomb

- ***bomb.c*** **: C code of main routine**
- **To defuse your bomb(binary file), you need to disassemble the bomb**
- **You should keep track of disassembled code**
  - Using debugger
  - Using object dump tool



Disassemble code - objdump

- **Object dump**
  - `$> objdump -t bomb`
    - print the name of all functions & global variables & their addresses
  - `$> objdump -d bomb`
    - Use this to disassemble all of the code in the bomb
  - For more information, `man objdump`

Disassemble code - gdb

- **The GNU Project Debugger (GDB)**
  - Allows you to see what is going on inside the program while it executes
  - Can start your program, specifying anything that might affect its behavior
  - Can make your program pause on specified conditions
  - Can examine what has happened, when your program has stopped
  - Can change values in your program

- **Install GDB**
  - `$> sudo apt-get install gdb`

- **Run executable with GDB**
  - `$> gdb` *`nameOfExecutable`*
  - In this lab, *nameOfExecutable* is bomb

Disassemble code - gdb

- **Basic instructions**
  - `(gdb) run` : Start the program
  - `(gdb) continue` : Run the program until next breakpoint
  - `(gdb) breakpoint` : Make a breakpoint
  - `(gdb) delete` : Delete a breakpoint
  - `(gdb) step` : Run next line of code
  - `(gdb) next` : Run next line of code (not jumping into a function)
  - `(gdb) quit` : Quit gdb

- **Instructions for assembly code**
  - `(gdb) disassemble` : Disassemble the function / lines of code
  - `(gdb) stepi` : Run next line of assembly code
  - `(gdb) nexti` : Run next line of assembly code (not jumping into a function)

Disassemble code - gdb

- **Variable print instruction**
  - `(gdb) print` *func* : Print address of function *func*
  - `(gdb) p` *var* : Print value of variable *var*
  - `(gdb) p/[format]` *var* : Print value of *var* with format

    + format: t = binary, o = octal, d = int, u = unsigned int, x = hexadecimal, c = char, f = floating-point

- **Memory print instruction**
  - `(gdb) x/[range][format][unit]` *addr* : Print memory value

    + format: t = binary, o = octal, d = int, u = unsigned int, x = hexadecimal, c = char, f = floating-point, **s = string**, **i = assembly instr**

    + Unit: b = byte, h = halfword (2-byte), w = word (4-byte), g = giant word (8-byte)

Disassemble code - gdb

- **Information print instruction**
  - `(gdb) info registers` : Print all registers' value
  - `(gdb) info breakpoints` : Print all breakpoints

- **If you need more instruction detail**
  - `(gdb) help`
  - `$> man gdb`

Grading guideline

- **Each time your bomb explodes, it notifies our server**
- **You can get points by defusing each phase**
  - Phase 1-4: 15 points / Phase 5-6: 20 points
  - Total 100 points
- **You will lose your points when your bomb explodes**
  - -1 point per two explosion (maximum -20 points)
- **Due date is Oct 16th (Tue) 23:59 pm**
- **Cut-off date is Oct 19th (Fri) 23:59 pm**
  - We will shut down the grading server on time

Grading guideline

- **You can check your point on http://martini.snucse.org:15213/scoreboard**
  - This webpage is updated every 30 seconds

- **You can download new bomb repeatedly using same ID**
  - Your maximum point will be 100 although you defuse many bombs

Bomb Lab Scoreboard

This page contains the latest information that we have received from your bomb. If your solution is marked **invalid**, this means your bomb reported a solution that didn't actually defuse your bomb.

Last updated: Wed Sep 27 16:54:00 2017 (updated every 30 secs)

#	Bomb number	Submission date	Phases defused	Explosions	Score	Status
1	bomb1	Wed Sep 27 16:40	1	0	15	valid

Summary [phase:cnt] [1:1] [2:0] [3:0] [4:0] [5:0] [6:0] total defused = 0/1

Q&A

Reference

- http://visualgdb.com/gdbreference/commands/
- **http://www.yolinux.com/TUTORIALS/GDB-Commands.html**
- 유닉스 리눅스 프로그래밍 필수 유틸리티, 백창우, 한빛미디어
- **http://beej.us/guide/bggdb/**