Bomb Lab Recitation Notes (TA Guide)

Timings

Section	Timing
OH Etiquette + bomb lab overview	5 min
Registers + objdump + assembly reminders	7 min
GDB	7 min
Activity 1	13 min (give them 5-6 minutes to try it by themselves in groups, then spend 7 mins tracing through it with them; walk around during this time to help them and give them ideas on what to do in gdb)
Activity 3	13 min (give them 7-8 minutes to try it by themselves in groups, then spend 5-6 mins tracing through it with them; walk around during this time to help them and give them ideas on what to do in gdb)
Questions	4 min

Important Notes

OH Etiquette

- Remind students about conceptual OH
- Emphasize what sort of description they should put
- Encourage them to narrow the scope of their question and debug before OH as TA's can only stay with them for 10-u12 min!

Assembly Reminders

- Remind students about the difference between constants, registers, and memory
- Maybe explain difference between caller-saved/callee-saved

Activity 1

- First show them the source code slide so they can see how many arguments main takes
- Tell them to look at the GDB cheat sheet part of the handout- NOT THE TRACES JUST YET

- Tell them to first look at the commands and see what they could do to start off with the activity- run, break points, printing arguments
- Separate them into groups of 3-4 (don't tell them, actually point to them and say how about you three-four work together, etc)
- Both TAs should walk around for 5-6 minutes and give each group some ideas on how to figure what the program is doing and what the assembly exactly shows them. Ask students to raise their hand once they figure out what the program is doing.
- Then go through the trace slides for 7-8 minutes and EXPLAIN each step
    - When you disas main, remind them that push $rbx is being done because rbx is callee-saved, so it must be stored before it is modified. The program is just printing out whatever argument you put in. The print (char*) 0x4ac6e8 should print out "%s\n"
    - You are printing argv[1] to show them that the argument they put is going into main
- Ask if they have questions

Activity 3

- Emphasize that this is more like the bomb lab assignment so they should be paying attention and really trying to figure this one out on their own
- Tell them to look at the GDB cheat sheet part of the handout- NOT THE TRACES JUST YET
- Tell them to first look at the commands and see what they could do to start off with the activity- run, break points, printing arguments
- Separate them into the same groups of 3-4
- Both TAs should walk around for 7-8 minutes and give each group some ideas on how to figure what the program is doing and what the assembly exactly shows them. Ask students to raise their hand once they figure out how to get "good args" to print.
    - Push them to look at the compare function because that's what tells the user if the arg is "good" or "bad"
- Then go through the trace slides for 5-6 minutes and EXPLAIN each step.
    - First look at cat act3.c. We are comparing two numbers and printing either good or bad args. So let's go to compare and see what its' doing.
    - print 0x3b6d → 15213. Ok so we are looking at the number 15213 somehow. Let's see what else is happening. We know there are two numbers stored in $rdi and $rsi. Its moving %rdi to %rbx then adding 5. Then, its adding $rsi to %rbx. So %rbx = %rdi + %rsi + 5. Then they are comparing that to 15213.
    - So what does this say about the two arguments we are inputting? (Now get them to tell you the answer)
- Ask if they have any questions

Questions

- Ask if they have questions!
- Tell them the reminders on the slide

More things to explain if there's time:

- Assembly and GDB syntax are different
    - %rax vs $rax
    - -0x4(%rax, %rdi, 4) vs $rax + $rdi * 4 - 0x4
- How to interpret cmpq followed by jl, etc.
- Show examples of how to print things in gdb
    - Formatting numbers
    - Formatting strings
    - Demonstrate printing the arguments to a function (print (int) $rdi)
- Demonstrate using other gdb commands?
    - stepi, nexti
    - backtrace, frame