EECS 678 - Lab 02

Theodore Lindsey

April 24, 2015

`http://people.eecs.ku.edu/~vvivekan/lab/gdb/gdb.html`

1. *What is wrong with the original code that eventually causes it to crash? (Obviously, the program fails becuase of an invalid memory reference producing a segmentation fault – what programming error led to the seg fault?)*

   In the `execute_simple_command` function, the following two instructions are issued:

   - `FREE (the_printed_command_except_trap);`
   - `the_printed_command_except_trap = the_printed_command;`

   .

   Unfortunately `the_printed_command_except_trap` and `the_printed_command` share the same address space and when `the_printed_command_except_trap` is freed, so is `the_printed_command`. When `the_printed_command_except_trap` is assigned the value of `the_printed_command` (which has also been already freed), the program segfaults.

2. *Describe how you diagnosed the problem with the original code. If you used GDB, which commands did you find most helpful? If you did not, what tools were most helpful in diagnosing the problem?*

   - Within `gdb`, we executed the command `r ../finder.sh ../bash-4.2/ execute 20` to run the script
   - When the program segfaulted, we checked the output of `backtrace` to determine where within the program the segfault occurred
   - We looked for the first source file that was actually within the source directory and identified the function that contained the error (`execute_simple_command`)
   - We issued the `up` command several times to load the correct frame associate with the problem function
   - We printed the contents of variables near to the segfaulting line to determine which variable was causing the problem (using `p ⟨variable name ⟩`)
   - We identified that both `the_printed_command_except_trap` and `the_printed_command` had a common address and that when one was "freed" the other was as well
   - We looked to other places where the `FREE` function was called and found that using the `savestring` would prevent this issue.

3. *Describe how your solution fixes the problem. Are you confident your solution is correct?*

   The `savestring` function gets around the problem of targeting a cleared address. Between comparing the modification to similar cases within the source code and testing the compiled version, I'm certain that I found the correct fix.

   The change that was made was from (a) to (b).

   (a) `the_printed_command_except_trap = the_printed_command;`

   (b) `the_printed_command_except_trap = savestring (the_printed_command);`