Sample Project 2 Write-up

Jonathan Misurda
jrmst106@pitt.edu

1 Lab Program

1.1 Procedure

Since the programs are all written in C, they will have a `main` function, so I initially placed a breakpoint at `main` and did a disassembly. I noticed there were many `call` instructions and `gdb` showed that one in particular was a `call` to `strcmp`. Thinking that this was probably where the input was compared with the solution, I placed a breakpoint here. I examined the contents of the registers that had been moved to the stack, knowing that these will be the parameters according to the calling convention we have discussed in class.

The first parameter I examined proved to be the string "bcdefg". I tried to enter this as my solution string, but was told this wasn't correct. I went back to the breakpoint at `strcmp` and saw the "bcdefg" as I had seen before, but the second parameter was "cdefgh" which was not the string I had entered. The code between the input and the comparison was changing the string in some way.

1.2 Solution

I reasoned that the transformation was simply that every letter was being increased by one. I entered the string "abcdef" and was told that the solution was "bcdefg".

1.3 Post-Mortem/Notes

- By the large size of the file and the fact that gdb was able to resolve the calls to their library function names, the executable was most likely statically linked and had the symbol table left intact.
- The strings command would have shown the "bcdefg" constant in the program because it resides in the string table of the executable. This is a result of the string being a literal inside the program.
- It is easy to recognize arguments by small negative offsets of `$esp` and parameters by positive offsets of `$ebp`.