CSE 127: Computer Security (Fall 2018) **Due October 4, 10:00 P.M. PDT**

Assignment 1

20 pts

The goal of this assignment is to become familiar with the setup that will be used for future projects, such as the use of a virtual machine and the turn-in script. To this end, you will complete a few simple excercises involving the GNU debugger (GDB), which you will be using for the next assignment. You will be provided with a file to debug and a text file to fill out with your answers. Your solution is due on October 4, 10:00 P.M. PDT. You may work with *one* other person in your class section on the assignment; if so you should only submit one solution for the two of you.

1 Getting Started

To complete this assignment you will be provided with a VirtualBox VM and a set of files including a turn-in script.

1.1 VM Image

In order to match the environment in which your submission will be graded, all work for this assignment must be done on the VirtualBox VM we have provided named `cfbox`. You can download the VM image from:

https://drive.google.com/file/d/1IGTFd60VijSqUw3r_pOSBBSj5-qt02-7/view?usp=sharing

The VM is configured with two users: `student`, with password "`hacktheplanet`", and `root` with password "`hackallthethings`". The VM is configured with SSH on port 2222. Please note that SSH is disabled for `root`, so you can only SSH in as the student user. You can still log in as `root` using `su` or by logging into the VM directly.

To SSH into the VM:

```
ssh -p 2222 student@127.0.0.1
```

To copy files from your computer to the VM:

```
scp -P 2222 -r /path/to/files/ student@127.0.0.1:/home/student
```

To copy files from the VM to your computer:

```
scp -P 2222 student@127.0.0.1:/path/to/files/ /destination/path
```

1.2 Submission Script

To submit your solution, use the turn-in script provided with the starter files. It will archive your submission and submit it to our server to be graded. You may submit multiple times, but only your latest submission will be graded. Fill out the `PID` file with your PID on the first line. If you're working with a partner, include both PIDs on the first line separated by a space.

1.3 Assignment Files

You will be provided with starter files for the assignment on the class webpage, including a `fib.c` to debug and a `hw1.txt` to fill out with your answers. To build `fib`, run `make` on the command line.

2 Using GDB

To run the executable in GDB, run `gdb -e fib`. Use the `-s` option to load a symbol file. I recommend the following workflow in GDB:

1. **Starting**. Set breakpoints that you can later use for analysis:
   - `b foo` — break at function `foo`
   - `b *0x08048489` — break at the instruction at address 0x08048489
   - `r` — run the executable
2. **Analyzing**. Examine memory, registers, etc.; disassemble code; show stackframe, backtrace, etc; and more:
   - `disas foo` — disassemble function `foo`
   - `i r` — view registers
   - `x <loc>` — examine memory
   - `x $eip` — examine current instruction pointer
   - `x $ebp+4` — examine return address
   - `x /10x $esp` — examine 10 words at top of stack
   - `x /10x buf` — examine 10 words in `buf`
   - `x /10i $eip` — examine 10 instructions starting at instruction pointer
   - `x /10i buf` — examine 10 instructions starting at `buf`
3. **Continuing**. Continue analysis:
   - `c` — continue execution until next breakpoint/watchpoint
   - `si` — step to the next instruction
   - `s` — step to the next line of source code

Note that this is only a cursory overview of GDB; much more info is available from online resources.

3 Assignment Instructions

Complete the following excercises and fill out `hw1.txt` with your answers. Give numerical values in hex unless otherwise stated.

1. What is the value of the `ecx` register when function `f` is called?
2. Which register stores the value of `n` when `f` is called?
3. What is the address of `f`?
4. What is the 6th instruction of `f`?
5. What address does `f` return to in `main`?