Popa & Wagner
Spring 2016

CS 161 Computer Security — Homework 1

Due: Monday, February 1st, at 11:59pm

Instructions. Create an EECS instructional class account if you have not already. To do so, visit `https://acropolis.cs.berkeley.edu/~account/webacct/`, click "Login using your Berkeley CalNet ID," then find the cs161 row and click "Get a new account." Be sure to take note of the account login and password.

Log in to your instructional account, create an empty directory, and create files `q1.txt`, `q2.txt`, `q3.txt`, and (optionally) `q4.txt` in that directory following the instructions below. To submit the first part of your homework solution, type `submit hw1` from that directory. If you realize you missed anything, you can always submit again up until the submission deadline.

For Q5, please write your answers on a separate sheet of paper. Put your name and student ID at the top, and submit it using Gradescope. You should have received an email with an invitation to Gradescope; use that account to submit Q5. Q5 is not optional—don't forget to submit your answer to Q5 on Gradescope.

This homework is due **Monday, February 1st, at 11:59pm**. It *must* be submitted electronically. No late homeworks will be accepted.

Problem 1 ***Policy*** **(10 points)**

The aim of this exercise is to ensure that you read the course policies, as well as to make sure that you are registered in the class and have a working EECS instructional class account.

Open the course website `http://www-inst.eecs.berkeley.edu/~cs161/sp16/`.
Append `?lastname=<name>&userid=cs161-xy` to the URL in the address bar, where `<name>` is your last name (as in campus records) and `cs161-xy` is your class ID (with `xy` replaced with the two final characters of your class account). (If you have spaces or apostrophes in your last name, go ahead and type them in: they should not cause any problems.) Thus, the URL you will open is:

`http://www-inst.eecs.berkeley.edu/~cs161/sp16/?lastname=<name>&userid=cs161-xy`

Please read and check that you understand the course policies on that page. If you have any questions, please ask for clarification on Piazza.

To receive credit for having read the policies, create a text file called `q1.txt` and put in it the string "`I understand the course policies.`" (no quotes necessary)

Problem 2 ***Policy*** **(10 points)**

You're working on a course project. Your code isn't working, and you can't figure out why not. Is it OK to show another student (who is not your project partner) your draft code and ask them if they have any idea why your code is broken or any suggestions for how to debug it?

Read the course policies carefully. Based on them, determine the answer to this question. Create a text file called `q2.txt` and put your answer to this question in it. A one-word answer is fine: we do not need a detailed explanation (though you may provide an explanation if you choose).

Problem 3 ***Hidden Password*** **(30 points)**

We have hidden a password on the web page you visited in Question 1. If you entered the URL correctly as above (substituting your last name and userid), you'll be able to access the password—if you know the trick.

What's the trick? You're going to have to figure that out on your own. But we'll help you out: we will share a hint on Piazza. Make sure you have set up your Piazza account (using the same name on Piazza as the University has for you in its records), which you can do by following the link provided on the course web page. Then, go look for the question on Piazza where we disclose the hint.

Once you have figured out the password, create a text file called `q3.txt` and put a single line in it whose contents are just the password itself. In accordance with the class policies on doing work individually, do not share the password with anyone.

Problem 4 ***Feedback*** **(0 points)**

Optionally, feel free to include feedback. What's the single thing we could do to make the class better? Or, what did you find most difficult or confusing from lectures or the rest of class, and what would you like to see explained better? If you have feedback, create a text file called `q4.txt` with your comments.

Problem 5 ***Memory Layout*** **(50 points)**

Consider the following C code:

```
 1 int frog(int i, int j, int k) {
 2     int t = i + j;
 3     char *buf = malloc(2);
 4     t = t + k;
 5     return t;
 6 }
 7
 8 void duck(char *buffer) {
 9      int i[2];
10      i[0] = 1;
11      i[1] = frog(i[0], 2, 3);
12 }
13
14 int main() {
15      char foo[4];
16      duck(foo);
17      return 0;
18 }
```

The code is compiled and run on a 32-bit x86 architecture (i.e., IA-32). Assume the program is run until line 5, meaning everything before line 5 is executed (i.e., a breakpoint was set at line 5). We want you to sketch what the layout of the program's stack looks like at this point. In particular, print the template provided on the next page and fill it in. Fill in each empty box with the value in memory at that location. Put down specific values in memory, like 1 or 0x00000000, instead of symbolic names, like foo. Also, on the bottom, fill in the values of %ebp and %esp when we hit line 5.

Assumptions you should make:

- memory is initially all zeros
- execution starts at the very first instruction in main(), and %esp and %ebp start at 0xc0000000 at that point
- the call to malloc returns the value 0x12345678
- the address of the code corresponding to line 17 is 0x1111110
- the address of the code corresponding to line 12 is 0x1111144
- a char is 1 byte, an int is 4 bytes
- no function uses general-purpose registers that need to be saved (other than %ebp)

See the next page for the template.

0xbffffffc:	
0xbffffff8:	
0xbffffff4:	
0xbffffff0:	
0xbfffffec:	
0xbfffffe8:	
0xbfffffe4:	
0xbfffffe0:	
0xbfffffdc:	
0xbfffffd8:	
0xbfffffd4:	
0xbfffffd0:	
0xbfffffcc:	
0xbfffffc8:	
	⋮

%ebp =

%esp =