Paxson Spring 2017 — CS 161 Computer Security — Homework 1

Due: Thursday, January 26, at 11:59pm

Instructions. This homework is due **Thursday, January 26, at 11:59pm**. No late homeworks will be accepted. This assignment must be done on your own.

Create an EECS instructional class account if you have not already. To do so, visit https://inst.eecs.berkeley.edu/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, and log in to your instructional account.

Make sure you have a Gradescope account and are joined in this course. The homework *must* be submitted electronically via Gradescope (not by any other method). Your answer for each question, when submitted on Gradescope, should either be a separate file per question, or a single file with each question's answer on a separate page.

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 https://www.icir.org/vern/cs161-sp17/.
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:

https://www.icir.org/vern/cs161-sp17/?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, submit the following statement as your answer for Q1: "`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, and submit it for Q2 on Gradescope. 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 Backdoor*** **(30 points)**

We have hidden a secret password (a "backdoor") 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 post on Piazza where we disclose the hint.

Once you have figured out the password, submit just the password as your answer to Q3 on Gradescope. 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, submit your comments as your answer to Q4.

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

Consider the following C code:

```
1  int dog(int i, int j, int k) {
2      int t = i + k;
3      char *newbuf = malloc(4);
4      t = t + 4;
5      t = t - j;
6      return t;
7  }
8
9  void cat(char *buffer) {
10     int i[2];
11     i[1] = 5;
12     i[0] = dog(1, 2, i[1]);
13 }
14
15 int main() {
16     char buf[8];
17     cat(buf);
18     return 0;
19 }
```

The code is compiled and run on a 32-bit x86 architecture (i.e., IA-32). Assume the program is run until line 6, meaning everything before line 6 is executed (i.e., a breakpoint was set at line 6). 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 buf. Also, on the bottom, fill in the values of %ebp and %esp when we hit line 6. Submit the filled template as your answer to Q5 on Gradescope.

Assumptions you should make:

- memory is initially all zeros
- execution starts at the very first instruction during the usual invocation of main(), and %esp and %ebp start at 0xa0000064 at that point
- the call to malloc returns the value 0x12341234
- the address of the code corresponding to line 4 is 0x01111180
- the address of the code corresponding to line 13 is 0x01111134
- the address of the code corresponding to line 18 is 0x01111100
- 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.


0xa0000060:
0xa000005c:
0xa0000058:
0xa0000054:
0xa0000050:
0xa000004c:
0xa0000048:
0xa0000044:
0xa0000040:
0xa000003c:
0xa0000038:
0xa0000034:
0xa0000030:
0xa000002c:
0xa0000028:
0xa0000024:
0xa0000020:
0xa000001c:
0xa0000018:
0xa0000014:
% ebp =
% esp =