Popa & Weaver Spring 2019 — CS 161 Computer Security — Homework 1

Due: Tuesday, 5 February 2019, at 11:59pm

Instructions. This homework is due on **Tuesday, 5 February 2019, at 11:59pm**. No late homeworks will be accepted unless you have prior accomodations from us. 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 be 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 http://inst.eecs.berkeley.edu/~cs161/sp19/.

Read the course policies and answer the following question:

How many project "slip days" do you get?

There are no "slip days".

Problem 2 ***Collaboration*** **(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?

Select if yes ☐

Problem 3 ***Security Principles*** **(20 points)**

For each of the following paragraphs, there is exactly one security principle that best applies to the situation described. Select the best **four options** from below after reading the following scenario:

Getting on the cryptocurrency hype, one day Bob decides to set up his own exchange. He sets up all the infrastructure, but worries about forgetting the password, so Bob hides his login credentials in an HTML comment on the login page.

Eventually, Bob manages to gather a large user-base and realizes his site looks like a back-end developer trying to learn CSS, so he contracts out front-end work to Mallory's Do-No-Evil design firm (for an incredible price too!). He gives them an account with access to his front-end and back-end codebase, and databases of user information as well.

Finally, Bob wants to enforce password security. Bob requires every user to use a "super-secure" password; that is, the password cannot contain any English word, cannot contain any birthday, and must have many special characters (e.g., $ %). The user needs to type in this password every 5 minutes. Bob disables the clipboard on the password field; in this way, the user must manually enter the password, nothing else.

Unfortunately for him, one day he wakes up to his website being featured on a well-known news site after a data leak. Pressured by an internet mob, he hires a contractor to find all the issues with his site. However, fixing the website ended up being a different story, as much of the code was written (uncommented) in a late-night coffee-fuled frenzy, and Bob finds that he can't change any aspect of the website without breaking it in its entirety. In a panic, Bob announced the closure of his site and goes into hiding.

1. Does Bob violate **security is economics**? Select if yes [ ]

2. Does Bob violate **least privilege**? Select if yes [X]

3. Does Bob violate **fail-safe defaults**? Select if yes [ ]

4. Does Bob violate **separation of responsibility**? Select if yes [ ]

5. Does Bob violate **don't rely on security by obscurity**? Select if yes [X]

6. Does Bob violate **consider human factors**? Select if yes [X]

7. Does Bob violate **complete mediation**? Select if yes [ ]

8. Does Bob violate **detect if you can't protect**? Select if yes [ ]

9. Does Bob violate **design security in from the start**? Select if yes [X]

Problem 4 ***Vulnerable Code*** **(40 points)**

Consider the following C code:

```
1  void greet(char *arg)
2  {
3    char buffer[12];
4    printf(``I am the Senate. What is your name?\n");
5    scanf(``%s", buffer);
6    printf(``It's treason then, %s\n", buffer);
7  }
8
9  int main(int argc, char *argv[])
10 {
11   char beg[3] = 'Obi';
12   char end[11] = 'Wan Kenobi?';
13   strncat(beg, end, 5);
14   greet(argv[1]);
15   return 0;
16 }
```

1. What is the line number that has a memory vulnerability?

   5

2. What is this vulnerability called?

   Buffer overflow

3. Just before the program executes the line in part 1, the registers are:

   `%esp: 0xBFFFF820` `%ebp: 0xBFFFF848`

   Given this information, describe in detail how an attacker would take advantage of the vulnerability. Also make sure to include the address that the attacker needs to over-write. (Maximum 5 sentences)

   The attacker can input a very long string. The last 4 bytes should contain the malicious return address. This 4 bytes data will overwrite the original return address located at 0xBFFFF84C.

4. What would you change to fix the problem in part 1?

   Use fgets() function instead of scanf()

5. Given the code as is, would stack canaries prevent exploitation of this vulnerability?

   Select if yes [X]

   Why or why not?

   Canaries are random value that attacker doesn't know. If the attacker want to overwrite the return address, he has overwrite the canaries with a wrong value in the same time. Thus, by checking whether the value of canaries is unchanged, this attack can be detected and prevented.

Problem 5 ***Reasoning About Memory Safety*** **(35 points)**

Consider the following C code.

```
1  /* (a) Precondition: ------------------- */
2  void dectohex(uint32_t decimal, char* hex) {
3      char tmp[9];
4      int digit, j = 0, k = 0;
5      do {
6          digit = decimal % 16;
7          if (digit < 10) {
8              digit += '0';
9          } else {
10             digit += 'A' - 10;
11         }
12         /* (b) Invariant: --------------- */
13         tmp[j++] = digit;
14         decimal /= 16;
15     } while (decimal > 0);
16     while (j > 0) {
17         hex[k++] = tmp[--j];
18         /* (c) Invariant: --------------- */
19     }
20     hex[k] = '\0';
21 }
```

1. Please identify the **preconditions** that must hold true for the following code to be memory safe. In addition, the precondition must be as conservative as possible (e.g. `decimal` cannot be required to be solely zero). Justify why your given precondition cannot be any less strict.

   'hex' shouldn't be NULL and it should be a valid address. And the memory spaces it points to should be at least 9 bytes long to avoid possible overflow.
   Justify: 'decimal' is 32-bits unsigned integer number. So when it is converted into hexadecimal representation, it should be no longer than 8 character long. Thus, codes within the first do-while loop will be executed no more than 8 times. Plus 1 byte for null terminator in line 20. In total, at least 9 bytes is required to eliminate potential vulnerability.

2. Please identify the loop **invariants** (b, c) that must hold true and justify them as well.

   invariants(b): $0 < j < 8$ && $0 \leq$ decimal $\leq$ 0xFFFFFFFF && digit $\in$ {'0', '1', '2', ..., 'E', 'F'}
   'decimal' is 32-bits unsigned integer, therefore it will definitely greater than or equal to 0 and less than or equal to 0xFFFFFFFF. Since 'decimal' will be divided by 16 in each loop, this do-will loop will be executed no more than 8 times. This guarantee $0 < j < 8$. Line 7-11 make sure 'digit' will hold valid hexadecimal character.
   invariants(c): $0 < i < 8$ && $0 < j < 8$
   Since this while loop will be executed no more than 8 times.

Problem 6 ***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 Q6.