Computer Science 4271
Spring 2022
Midterm exam 1
February 22nd, 2022
Time Limit: 75 minutes, 9:45am-11:00am

- Before starting the exam, you can fill out your name and other information of this page, but don't open the exam until you are directed to start. Don't put any of your answers on this page.
- This exam contains 5 pages (including this cover page) and 4 questions. Once we tell you to start, please check that no pages are missing.
- You may use any textbooks, notes, or printouts you wish during the exam, but you may not use any electronic devices: no calculators, smart phones, laptops, etc.
- You may ask clarifying questions of the instructor or TAs, but no communication with other students is allowed during the exam.
- Please read all questions carefully before answering them. Remember that we can only grade what you write on the exam, so it's in your interest to show your work and explain your thinking.
- By signing below you certify that you agree to follow the rules of the exam, and that the answers on this exam are your own work only.

The exam will end promptly at 11:00am. Good luck!

Your name (print): ____________________

Your UMN email/X.500: ____________________`@umn.edu`

Number of rows ahead of you: __________ Number of seats to your left: __________

Sign and date: ____________________

Question	Points	Score
1	28	
2	20	
3	28	
4	24	
Total:	100	

1. (28 points) Multiple choice. Each question has only one correct answer: circle its letter.

   (a) Which of these `printf` format specifiers reads and prints a 64-bit value on Linux/x86-64?

   A. `%lx` B. `%d` C. `%c` D. `%%` E. `%x`

   (b) Which of these mechanisms primarily exists to stop repudiation threats?

   A. ASLR B. W$\oplus$X C. encryption D. logging E. passwords

   (c) Which of these values is **not** found in the stack on Linux/x86-64?

   A. A global variable
   B. An environment variable
   C. A local array variable in a function
   D. A function return address
   E. The program's command-line arguments

   (d) In the architecture of our system, software component A requests some information from component B and gets a response. What arrowheads should we draw on the edge connecting boxes for A and B?

   A. Neither, because arrowheads make the diagram cluttered.
   B. Only from A to B, because the response could not be part of a threat.
   C. Only from B to A, because the request could not be part of a threat.
   D. Both, because both the request and the response could be part of a threat.

   (e) Which of these is **not** a kind of code-reuse attack?

   A. Call-oriented programming
   B. Jump-oriented programming
   C. Return-to-libc
   D. Integer overflow
   E. Return-oriented programming

   (f) Which of these is **not** either a synonym for a kind of defense, or a hardware mechanism that implements it?

   A. W$\oplus$X B. NX bit C. DEP D. ASLR E. XD bit

   (g) A buffer overflow within a `struct` located on the heap might overwrite all of the following **except**:

   A. A field located later in the same structure
   B. A pointer used to locate another heap object
   C. A field in another heap object
   D. A local variable
   E. An area that will be reused in a later heap allocation

2. (20 points) C programming bugs.

   Each of the following snippets of C code illustrates an erroneous code pattern that might cause a program to crash. For each code example, give a value of the variable `x` that would cause the dangerous behavior, and the name of the bug. (Assume that `malloc` never fails in these examples.)

   (a)
```
char *p = malloc(10);
if (x >= 10)
    free(p);
if (x < 11)
    free(p);
```

   Value of x: Bug name:

   (b)
```
int a[20], i;
for (i = 0; i < x; i++)
    a[i] = 12;
```

   Value of x: Bug name:

   (c)
```
char *p = malloc(20);
if (x < 20)
    free(p);
p[0] = p[1];
```

   Value of x: Bug name:

   (d)
```
*x = 30;
```

   Value of x: Bug name:

   (e)
```
printf(x, 36, "carrot");
```

   Value of x: Bug name:

3. (28 points) Buffer overflow with machine code.

   The following function from a Linux/x86-64 program has a buffer overflow vulnerability. (It uses the function `strlcpy` which is like `strcpy` but taking a maximum size parameter. Unfortunately the maximum size parameter is set too large.)

   Answer the following questions about details of how the code was compiled that relate to how the vulnerability could be attacked. Write numbers in your answers in decimal. Some of your answers should be expressed in relation to the variable R, which represents the value held in both the `%rbp` and `%rsp` registers after the instruction numbered `2:`. These answers should be something like $R + 17$ or $R - 4000$.

   Below are excerpts of the relevant code in C and assembly language.

```
void f(char *p) {
    char buf[32];
    strlcpy(buf, p, MAX_SIZE);
}
```

```
 1:   push   %rbp
 2:   mov    %rsp,%rbp
 3:   sub    $0x30,%rsp
 4:   mov    %rdi,-0x28(%rbp)
 5:   mov    -0x28(%rbp),%rcx
 6:   lea    -0x20(%rbp),%rax
 7:   mov    $0x50,%edx
 8:   mov    %rcx,%rsi
 9:   mov    %rax,%rdi
10:   call   strlcpy
11:   leave
12:   ret
```

   (a) What is the location of the function's return address, relative to R?

   (b) What is the starting location of the buffer `buf`, relative to R?

   (c) What is the value of the third argument to `strlcpy` (i.e., of the macro `MAX_SIZE`)?

   (d) If the bytes within the buffer are numbered starting at 0, what is the range of byte values that would overwrite the return address?

4. (24 points) Short answers. Each of the following questions has a brief answer.

   (a) An `unsigned short` variable in C is 16 bits long, and holds a value between 0 and 65535. Suppose that `us16` is an `unsigned short` containing 16. Give, in hexadecimal, a value for an unsigned short `usx` such that `us16 * usx < usx`, according to the arithmetic rules of C.

   (b) What is the name for the kind of picture with boxes and arrows we use to describe the structure of a system in threat modeling?

   (c) The functions `strcpy` and `memcpy` both copy memory regions, but an important difference is that `strcpy` treats one particular byte value differently; what is it?

   (d) In 2015, it was revealed that computer systems of the US government's Office of Personnel Management (OPM) had been attacked over the previous years, apparently allowing the attackers to obtain information about millions of government employees such as fingerprints and security clearance forms. If the OPM had performed threat modeling under the STRIDE taxonomy, what kind of threat was the possibility of this kind of attack?

   (e) Trying to access memory via a pointer that is outside any valid memory region will cause a program to immediately crash with a segfault. If this is the only dangerous behavior that a bug allows, the STRIDE threat it is most relevant to is:

   (f) The instructions in a NOP sled don't do anything. So why is a NOP sled useful to an attacker?