Computer Science 5271
Fall 2019
Midterm exam
October 21st, 2019
Time Limit: 75 minutes, 1:00pm-2:15pm

- This exam contains 8 pages (including this cover page) and 4 questions. Once we tell you to start, please check that no pages are missing.
- 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.
- 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 TA, 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 2:15pm. 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	30	
2	24	
3	26	
4	20	
Total:	100	

1. (30 points) Multiple choice. Each question has only one correct answer: circle its letter.

   (a) This character is used as part of a printf format specifier to control what argument a value is taken from:

   A. `@` B. `$` C. `0` D. `s` E. `-`

   (b) The pathname passed to the Unix open system call cannot contain this character:

   A. `/` B. `\0` C. `\` D. `*` E. space

   (c) This security design idea is used in both qmail and Android:

   A. Using Unix UIDs to isolate code instead of human users
   B. Using C++'s `std::string` class exclusively
   C. Requiring add-on software to be cryptographically signed
   D. Sandboxing code from the Internet using SFI
   E. Having only a single developer

   (d) Which of these values would commonly **not** change under ASLR?

   A. The address of an environment variable
   B. The address of `main`'s stack frame
   C. The relative distance between `main` and `printf`
   D. The relative distance between a return address and a local variable
   E. The address of `printf`

   (e) "Heartbleed" was the name of a high-profile vulnerability disclosed in 2014. It was caused by faulty bounds checking, but because the unsafe access was a read instead of a write, the only kind of security policy that was directly violated was:

   A. confidentiality B. availability C. authentication D. integrity

   (f) Under a CFI implementation with two equivalence classes for calls and returns, and no shadow stack, an attacker could still redirect a function return to:

   A. A gadget that starts in the middle of an intended instruction
   B. Shellcode in an environment variable
   C. A call-preceded gadget
   D. The `system` function in the C library, always
   E. The `system` function in the C library, but only if its address was taken

(g) Password capabilities are similar to passwords in that:

A. They are chosen by users
B. To be secure, they must have high entropy
C. They are gradually being replaced by fingerprints
D. They are vulnerable to dictionary attacks
E. To be secure, they must be changed frequently

(h) One feature commonly added to high-security operating systems is tamper-proof logging. Which of Saltzer and Schroeder's design principles is this an instance of?

A. compromise recording
B. separation of privilege
C. confidential reservation
D. separation of powers
E. separation of duty

(i) Which of these CPU features does **not** lead to transient execution?

A. single-instruction multiple-data (SIMD) instructions
B. branch target prediction
C. return stack buffer usage
D. delayed memory exceptions
E. branch direction prediction

(j) At the University of Minnesota, you type in the same username and password to log into CSE Labs workstations, to access the WiFi network, and to view library resources off campus. This is an example of:

A. two-factor authentication
B. biometric authentication
C. economy of mechanism
D. centralized authentication
E. single sign-on

2. (24 points) Social media reliability relations

A commonly-raised concern about social media is that it is sometimes used to propagate unreliable information. Inspired by the use of lattices to define integrity policies in multi-level secure systems, your friend Parker is considering applying a similar mechanism to deal with unreliable information in Twitter.

(a) Parker's basic idea is to define an ordering relation $\sqsubseteq$ based on reliability. $B \sqsubseteq A$ will hold when user A is more reliable than B; only if this is the case will B be allowed to read A's tweets. On the left are some mathematical properties that the relation $\sqsubseteq$ might have. Match them with the intuitive descriptions on the right of what the properties mean in this context: each one is used exactly once.

A. If $A \sqsubseteq B$ and $B \sqsubseteq C$, then $A \sqsubseteq C$

B. $A \sqsubseteq A$

C. If $A \sqsubseteq B$ and $B \sqsubseteq A$, then $A = B$

D. Either $A \sqsubseteq B$ or $B \sqsubseteq A$

____ Everyone can read their own tweets

____ If two users are each at least as reliable as the other, they are equally reliable

____ For any two people, at least one can read the other's tweets

____ If you can read a retweet, you can also read the original tweet

(b) Keeping with the spirit of social networks, Parker has decided that reliability will be defined based on popularity, specifically by a pair of non-negative integers (t, i) where t is the number of Twitter followers a user has, and i is their number of Instagram followers. (t or i is also 0 if the user doesn't have Twitter or Instagram account at all.) But there was still some disagreement about the exact definition.

Leslie believes that Twitter is a better indication of reliability than Instagram, and proposed the following definition:

$$(t_1, i_1) \sqsubseteq_L (t_2, i_2) \iff t_1 < t_2 \vee (t_1 = t_2 \wedge i_1 \leq i_2)$$

Does Leslie's definition satisfy all four properties A-D mentioned above? If not, choose one property and give an example of a situation in which it does not hold.

(c) Chris thought it was more important for the definition to treat Twitter and Instagram equally, and proposed the following definition:

$$(t_1, i_1) \sqsubseteq_C (t_2, i_2) \iff t_1 \leq t_2 \wedge i_1 \leq i_2$$

Does Chris's definition satisfy all four properties A-D mentioned above? If not, choose one property and give an example of a situation in which it does not hold.

(d) Dakota liked Chris's suggestion, but wanted a definition that allowed more tweets to be read, and so proposed the following:

$$(t_1, i_1) \sqsubseteq_D (t_2, i_2) \iff t_1 \leq t_2 \vee i_1 \leq i_2$$

(This is the same as Chris's, but with "or" instead of "and".) Does Dakota's definition satisfy all four properties A-D mentioned above? If not, choose one property and give an example of a situation in which it does not hold.

(e) Which of the definitions $\sqsubseteq_L$, $\sqsubseteq_C$, and/or $\sqsubseteq_D$, form(s) a lattice?

(f) Before Parker and his friends visit California to talk to venture capitalists, is there any other problem with or objection to their idea they should consider?

3. (26 points) Return-to-libc attack.

   You are trying to carry out an attack against a Linux/x86-32 program that has a buffer overflow vulnerability. In your preliminary research, you've found that the vulnerability is completely permissive as to the data used in the overflow (even `\0` bytes are OK), and ASLR is disabled. But the bad news is that it only lets you write 44 bytes into a 32-byte buffer, which is not enough to overwrite a return address.

   Your backup plan is to carry out a return-to-libc attack, to let you make the program do the equivalent of `system("/tmp/evil")` using code that already exists in the binary. (You have already prepared the script `/tmp/evil`.) But you have to figure out how to make that happen by overwriting just the data that the overflow reaches. Below we've shown the C and assembly code for the relevant parts of the vulnerable program. On the next page, we've shown a picture of the area of memory that holds function `g`'s stack frame. For the locations that you control, there are blank spaces for you to fill in the data you want to go in those locations. Each blank space represents one byte: fill it in with either a single ASCII character like `A` or two hex digits like `ff`. You may leave a blank empty if it is not needed for your attack.

   The address of the function `system` is `0x0804f170`. It's not too important what the program does after calling `system`, but if you'd like to have it call `exit`, the address of that function is `0x804e670`.

   The blank spaces for the bytes are in order of increasing address left to right. So be careful about the order in which you write things, for instance depending on if they are a string or an address. (x86 is little-endian, putting the least-significant byte of a word at the lowest address.)

```
unsigned char input[44];
void f(char *str) {
    int len =
      parse(str, input, sizeof(input));
    g(input, len);
}

void g(unsigned char *data, int len) {
    unsigned char buf[32];
    memcpy(buf, data, len); /* oops */
    return;
}
```

```
f:
    /* ... */
    call   g
    add    $0x10,%esp
    mov    %ebp, %esp
    pop    %ebp
    ret

g:
    push   %ebp
    mov    %esp,%ebp
    sub    $0x28,%esp
    mov    0xc(%ebp),%eax
    sub    $0x4,%esp
    push   %eax
    pushl  0x8(%ebp)
    lea    -0x28(%ebp),%eax
    push   %eax
    call   memcpy
    add    $0x10,%esp
    mov    %ebp, %esp
    pop    %ebp
    ret
```

Address	Original purpose	Overwrite with
0xffffc3d4:	g arg 2 (len)	
0xffffc3d0:	g arg 1 (data)	
0xffffc3cc:	g→f return address	
0xffffc3c8:	saved %ebp	____ ____ ____ ____
0xffffc3c4:	unused	____ ____ ____ ____
0xffffc3c0:	unused	____ ____ ____ ____
0xffffc3bc:	buf[0x1c] – buf[0x1f]	____ ____ ____ ____
0xffffc3b8:	buf[0x18] – buf[0x1b]	____ ____ ____ ____
0xffffc3b4:	buf[0x14] – buf[0x17]	____ ____ ____ ____
0xffffc3b0:	buf[0x10] – buf[0x13]	____ ____ ____ ____
0xffffc3ac:	buf[0x0c] – buf[0x0f]	____ ____ ____ ____
0xffffc3a8:	buf[0x08] – buf[0x0b]	____ ____ ____ ____
0xffffc3a4:	buf[0x04] – buf[0x07]	____ ____ ____ ____
0xffffc3a0:	buf[0x00] – buf[0x03]	____ ____ ____ ____

Some more hints:

- After returning from memcpy, %esp contains 0xffffc3a0 and %ebp contains 0xffffc3c8.
- In AT&T syntax, the destination is the final operand, so for instance mov %eax, %ebx copies the contents of %eax into %ebx.
- The arguments to functions on Linux/x86-32 are passed on the stack, with the first argument at the lowest address. For instance, the relative position of the arguments and return address shown for g is also the same for other functions.
- system will need to create its own stack frame, so make sure that doesn't overwrite anything needed for your attack.
- You can assume that all of the stack addresses will be the same each time the program runs.

4. (20 points) Matching definitions and concepts. Fill in each blank with the letter of the corresponding answer. Each answer is used exactly once.

   (a) ____ Common Orange Book level for enhanced Unix variants

   (b) ____ Uses ambient authority for undesirable actions

   (c) ____ Trusted code that checks all sensitive operations

   (d) ____ A scripting language based largely on strings

   (e) ____ System call used to drop privileges

   (f) ____ Used to implement debugging and system call interposition

   (g) ____ An isolated environment for untrusted code

   (h) ____ Library function to reload register state

   (i) ____ Used on directories with multiple independent users

   (j) ____ The absence of a connection between systems

   (k) ____ An attack against kernel isolation using transient execution

   (l) ____ A fuzzing tool that incorporates coverage feedback

   (m) ____ A number identifying a subject in Unix access control

   (n) ____ Will abort instead of overflowing a buffer

   (o) ____ False positive and negative rate in a balanced configuration

   (p) ____ A common mistake in dynamic memory management

   (q) ____ Like a NOP sled for ROP

   (r) ____ A filesystem-only isolation mechanism

   (s) ____ An attack that is possible without an account

   (t) ____ Used to prevent precompuation of password hashes

   A. AFL B. air gap C. C2 D. `chroot` E. confused deputy F. double free G. EER H. `longjmp` I. Meltdown J. `ptrace` K. reference monitor L. remote exploit M. ret2pop N. salt O. sandbox P. `setresuid` Q. sticky bit R. `strcat_s` S. Tcl T. UID