

This practice midterm contains 9 pages (including this cover page) and 5 questions. The total number of points is 92 (these points do not count towards your grade and you do not have to turn in this practice midterm; they are added only to give you a feel for what the real midterm might look like). You are free to work on this practice midterm with other students in the class. The midterm will also include the academic integrity statement found below. You will be signing and acknowledging your understanding of this statement during the real midterm.

Academic Integrity. The University of Pennsylvania’s Code of Academic Integrity states: “Since the University is an academic community, its fundamental purpose is the pursuit of knowledge. Essential to the success of this educational mission is a commitment to the principles of academic integrity. Every member of the University community is responsible for upholding the highest standards of honesty at all times. Students, as members of the community, are also responsible for adhering to the principles and spirit of the following Code of Academic Integrity.” You can find the principles at: <https://catalog.upenn.edu/pennbook/code-of-academic-integrity/>.

Grade summary table

Question	Points	Score
1	18	
2	20	
3	16	
4	22	
5	16	
Total:	92	

1. (18 points) **True or False**

For each of the following statements, state whether they are true or false by circling the appropriate label.

- (a) The stack in x86 grows from low memory addresses to high memory addresses.
A. True B. **False**
- (b) The results of a fuzzer can help inform the presence of vulnerabilities.
A. **True** B. False
- (c) A reasonable policy for memory page privileges is to set memory pages to be $W \oplus X$ (writable XOR executable).
A. **True** B. False
- (d) Breaking a Shift or a Vigenere cipher is easier when the ciphertext is smaller.
A. True B. **False**
- (e) Return-oriented programming is an effective mechanism that an attacker can use to bypass a non-executable stack defense.
A. **True** B. False
- (f) A computationally-bounded adversary cannot tamper with or modify a message encrypted with a one-time pad unless the adversary knows the secret key.
A. True B. **False**

2. (20 points) **Multiple choice questions**

The following is a set of multiple choice questions. For each question, circle **one** correct answer. Some questions require circling multiple responses, but they will make this explicit.

- (a) You are given a message m and its one-time-pad encryption c . Can you compute the one-time-pad key from m and c ?

- ☒ A. The key cannot be computed because a one-time pad is perfectly secure
- ☐ B. The key is $m \oplus c$
- ☐ C. Only half of the bits of the key can be recovered
- ☐ D. Only a quarter of the bits of the key can be recovered

- (b) Address space layout randomization is intended to prevent (**circle all that apply**):

- ☒ A. Buffer overflow attacks where code is placed on an executable stack
- ☐ B. Format string vulnerability attacks that reveal the location of data
- ☒ C. Return-oriented programming attacks
- ☐ D. Stack canaries from leaking sensitive information

- (c) In x86, the return value of a function is placed (**circle all that apply**):

- ☒ A. Always in the stack.
- ☐ B. In the heap if it larger than 1 memory word.
- ☐ C. Always in the `%eax` register
- ☒ D. In the `%eax` register if it fits in one memory word.

- (d) Recall the description of the `fork` and `execve` system calls from the man pages:

- `fork`: causes the creation of a new process. The new process (child process) is an exact copy of the calling process (parent process) except for process id and parent process id.
- `execve`: transforms the calling process into a new process. The new process is constructed from an ordinary file, whose name is pointed to by a path name.

In UNIX, a process P_1 can spawn a new process P_2 running a program “program2” by calling `fork()` and then `execve('`./program2`', ...)`. Which of the following statements is true?

- ☐ A. P_2 , which is running program2, inherits the permissions and privileges of the owner of program2.
- ☐ B. P_2 , which is running program2, inherits the permissions and privileges of process P_1 .
- ☐ C. P_2 , which is running program2, has the group permissions of P_1 and the user permissions of the owner of program2.
- ☐ D. The call to `execve` succeeds only if P_1 has the same UID as the owner of program2.

- (e) Consider the following sequence of UNIX commands that create a file called `a.txt`, change its permissions and ownership, and then try to remove it.

```
echo "hello" > a.txt
chmod 000 a.txt
sudo chown root a.txt
rm -f a.txt
```

The command `rm -f a.txt`:

- A. Fails because read/write/execute privileges were revoked
- B. Succeeds because `rm` acts on directories, not on files
- C. Fails because the user no longer owns the file
- D. Succeeds because the file still has the user's id as group id

3. (16 points) **Short answer questions.**

- (a) Suppose you are tasked with protecting bank account balances (this is the “asset”). Give two security properties that would make sense to provide to this asset. Define these properties generally, and what they mean in the context of the asset.

- (b) Assume a PRG $G : \{0, 1\}^s \rightarrow \{0, 1\}^n$ with the following property: Every 2nd bit is a 0 with probability $2/3$ and every 3rd bit is a 1 with probability $2/3$. Give one *good* statistical test for this PRG that outputs 1 when the input is likely random and 0 otherwise.

- (c) Recall that UNIX enforces file accesses via an access control matrix that consists of (R, W, X) privileges for (Owner, Group, Other). One drawback of this approach is that it makes it hard to express complex access control policies since one can only specify a single owner and group for each file. Suppose that we want the file “hello.txt” to be readable to the intersection of group1 and group2 (i.e., only to users who are part of both group1 and group2). Can this be implemented in UNIX without having to add all users in the intersection to a third group, group3? If no, say why not. If yes, give a concrete solution.

4. (22 points) **Buffer overflows.**

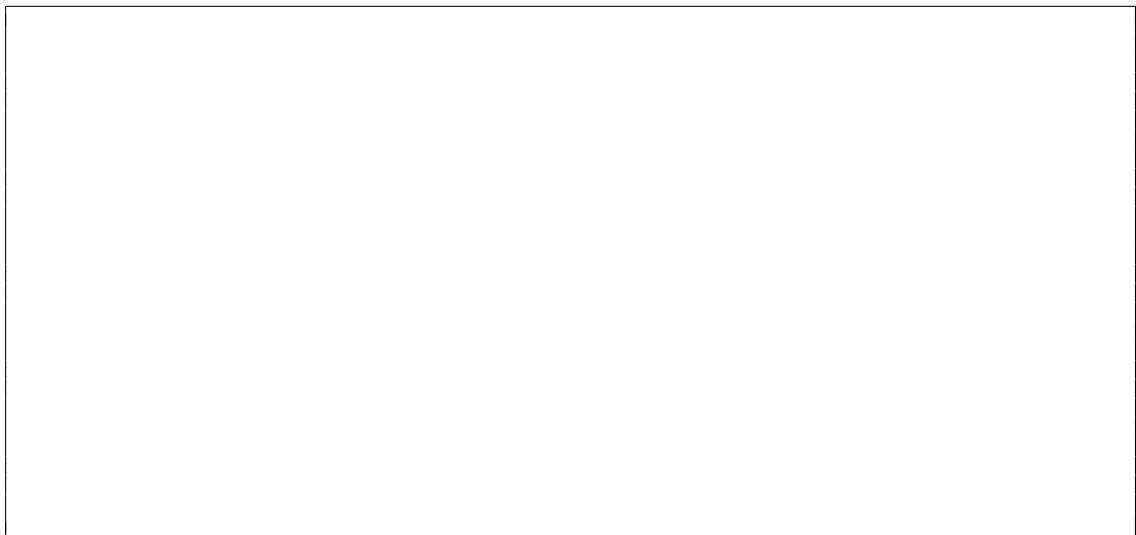
Consider the following snippet of C code. The values of X and Y are constants known at compile time (the compiler will resolve these constants before producing x86 assembly).

```
int vulnerable(char *arg) {  
    int a;  
    char buf[X];  
    strcpy(buf, arg);  
    a = Y;  
    return a;  
}
```

Consider the corresponding x86 assembly.

```
<+0>:  push %ebp  
<+1>:  mov  %esp,%ebp  
<+3>:  sub  $0x88,%esp  
<+9>:  mov  0x8(%ebp),%eax  
<+12>: mov  %eax,0x4(%esp)  
<+16>: lea  -0x6c(%ebp),%eax  
<+19>: mov  %eax,(%esp)  
<+22>: call 0x80481e0 <strcpy>  
<+27>: movl $0x2,-0x70(%ebp)  
<+34>: mov  -0x70(%ebp),%eax  
<+37>: leave  
<+38>: ret
```

- (a) Sketch the stack of the vulnerable function. For each of the following, show where in the stack they are located: (i) return address, (ii) arg, (iii) buf, (iv) a, (v) saved %ebp (it is sufficient to show their relative positions).



- (b) How big is the stack frame of this function (the number of bytes between the frame pointer and the stack pointer)? You can give this number in decimal or hexadecimal.

- (c) What value is the variable `a` set to (i.e., what is the value of `Y` in the C snippet)?

- (d) Assume that the compiler leaves an 8-byte gap between the `saved %ebp` and the next object in the stack. What is the size of `buf` (the value of `X` in the C snippet)? You can give the size in decimal or hexadecimal.

- (e) How large does the input provided in `arg` need to be to override the return address of the vulnerable function? You can give the size in decimal or hexadecimal.

5. (16 points) **One-time pad security** We will prove that for a cryptosystem to be perfectly secret, the key space (K) must be as large as the message space (M).

- (a) Give the definition of Shannon's perfect secrecy in terms of probability distributions. (Hint: the key is sampled uniformly from K .)

- (b) If $|K| = |M|$, what is the number of keys in K that map a given plaintext to a ciphertext?

- (c) Show that if $|K| < |M|$, the definition of perfect secrecy cannot be met.