Name: Nasser Alnasser Student ID: A18507030

CSE 127 Computer Security
Fall Quarter, 2025
Midterm Exam

Instructor: Stefan Savage

Name Nasser Alnasser
Student ID A18507030

Attention: This exam has twelve question sections and all of them count. Please write your name and student ID on each section. You have 80 minutes to complete the questions. As with any exam, you should read through the questions first and start with those that you are most comfortable with. Be sure to answer **all** parts of each question you answer. For each question, please **completely fill in the bubble** for your selection(s)

1	/10
2	/4
3	/4
4	/5
5	/5
6	/5
7	/5
8	/5
9	/4
10	/2
11	/4
12	/4
Total	/57

Name: Nasser Alnasser Student ID: A18507030

1. [10pts] True or False (**select (t) OR (f)**)

   a. Because the Unix passwd program is a `setuid` program owned by `root` it can only be executed by the `root` user.
   (t) True
   [✓ (f) False]

   b. Double-free vulnerabilities can be addressed by always overwriting an object with zeros (e.g., bzero(object, sizeof(object)) before it is freed.
   (t) True
   [✓ (f) False]

   c. ROP attacks exploit the fact that the most processors can't distinguish code from data and thus can be tricked into executing data from the attacker as though they were instructions.
   (t) True
   [✓ (f) False]

   d. The Meltdown side-channel attack would not be possible if the processor cache were disabled.
   [✓ (t) True]
   (f) False

   e. A 'NOP Sled" is used to prevent the return address on the stack from being overwritten.
   (t) True
   [✓ (f) False]

   f. Buffer overflows vulnerabilities are only a threat if they can overwrite control data on the stack (e.g., such as the return address or frame pointer)
   [✓ (t) True]
   (f) False

   g. Stack canaries (aka stack cookies) typically use random values so an attacker is not able to preserve the correct canary value during a stack buffer overflow that reaches the control data on the stack.
   [✓ (t) True]
   (f) False

   h. Format string vulnerabilities can arise from printf() statements that have fewer arguments than needed to satisfy the number of format descriptors in the format string.
   (t) True
   [✓ (f) False]

   i. Threat modeling is the approach the attacker uses to decide how to attack your system
   (t) True
   [✓ (f) False]

   j. The heap spray technique is used to bypass the Address Space Layout Randomization defense.
   [✓ (t) True]
   (f) False

Name: Nasser Almusser Student ID: A18507030

2. [4 pts] Security properties
   a. Recent news stories claim that foreign hackers have infiltrated key US telecommunication systems used to support court-ordered phone wiretaps and that, as a result, these hackers have been able to listen to the private phone calls from certain staffers in both presidential campaigns. If true, would this attack represent a violation of a CIA+P property and, if so, which one? (if you think multiple properties are implicated, pick the one that seems to be affected the most). **(select one)**
      [✓ Confidentiality]
      (b) Integrity
      (c) Availability
      (d) Privacy
      (e) None of the above

   b. If you visit to www.truepeoplesearch.com, you can look up the home address (and sometimes phone number and e-mail address) of virtually anyone. This database is populated via a combination of public records, scraping Web pages and purchasing of commercial data (e.g., mail-order retailers selling the collection of names and addresses of their customers). Does this capability represent a violation of a CIA+P property and, if so, which one? (if you think multiple properties are implicated, pick the one that seems to be affected the most). **(select one)**
      (a) Confidentiality
      (b) Integrity
      (c) Availability
      [✓ Privacy]
      (e) None of the above

In class, we discussed pin-tumbler locks as a metaphor for computer security. We demonstrated the "interface" for such locks being the insertion of an appropriately cut key into the keyway, which pushes the pins up in a manner that the space between top and bottom pins lines up precisely with the shear-line and the lock's "plug" can rotate and the lock can open. This mechanism assumes that only those in possession of this key, with its *particular secret* cuts, will be able to turn the lock. We then discuss a variety of ways in which this assumption is imperfect and how the locking mechanism can be attacked. For each of the attacks below, identify which of the CIA properties the attack violates *with respect to the lock itself.* (if you think multiple properties are implicated, pick the one that seems to be affected the most) **(select one for each question)**

   c. Manipulating the lock mechanism (e.g., raking, bumping, etc) to open it
      (a) Confidentiality
      [✓ Integrity]
      (c) Availability

   d. Filling the keyway with SuperGlue
      (a) Confidentiality
      (b) Integrity
      [✓ Availability]

Name: Nasser Alnasser Student ID: A18507030

3. [4 pts] In class, we discussed the attacker's use of NOP Sleds to avoid needing to know the precise address of a piece of shellcode. This is particularly true in the context of Heap Spray attacks where our goal is to maximize the chance that jumping into any address on the heap will lead to some copy of the shellcode. On Intel processors the NOP instruction is 0x90. One approach then to detect shellcode might be to look for long strings of 0x90 in input. Why might this not work well in practice? Give what you think is the most important reason in one sentence **(no more than twelve words)**

   Because there could be tons of Garbage strings of 0x90

4. [5 pts] In class, and in the Erlingsson et al paper ("Low-level Software Security"), we talked about the "return-into-libc" attack (an example of a "code reuse" attack) where control flow is redirected into an existing standard libc function. What is the attraction of this approach? **(select all that apply)**
   (a) It eliminates the requirement for a stack buffer overflow
   [✓ (b)] It implicitly bypasses ASLR because the address of the code segment cannot be randomized
   (c) It evades CFI because standard library functions are always callable from any call site.
   [✓ (d)] It evades DEP because standard library functions are in the code segment and not the stack or heap.
   (e) Because libc includes floating point functions, it allows integer overflow attacks that would not otherwise be possible

5. [5pts] Stack canaries are **(select all that apply)**:
   (a) Inserted and checked by code added by the compiler
   [✓ (b)] Inserted and checked by the operating system
   (c) Never cached to avoid introducing side channels
   [✓ (d)] Typically set to a random value, determined when the process starts
   (e) Redundant if DEP is implemented

6. [5pts] Hardware memory management, typically implemented via page tables, is commonly used to do which of the following? **(select all that apply)**
   [✓ (a)] Separate user processes into distinct and independent address spaces
   [✓ (b)] Protect the kernel's memory and code from being accessed by user processes
   (c) Prevent stack cookies from being overwritten
   (d) Detect NULL pointer accesses
   (e) Prevent heap meta-data from being corrupted

7. [5 pts] What are the *assumptions* under which hardware execution protection (aka DEP/W^X) is effective at stopping control flow integrity attacks? **(select all that apply)**
   [✓ (a)] The attacker is not able to write data to memory that is executable
   (b) The layout of memory is secret
   (c) The stack begins on a page boundary
   [✓ (d)] The attacker needs to execute code that they have injected into memory
   (e) The attacker is doesn't know the random value stored in the stack canary/cookie

Name: Nasser Alnasser Student ID: M850703 0

8. [5 pts] In class, we described an attack on a server where the attacker infers the stack canary value being used via repeated queries to the server. What *assumptions* does the attacker make in this attack? **(select all that apply)**
   (a) The same stack canary value is known
   (b) The stack base is known (i.e., no ASLR is being used)
   [✓ (c)] That when the runtime detects a stack cookie violation, it crashes the process and this can be distinguished from a request that doesn't crash the process
   (d) The stack canary value is random
   [✓ (e)] Every process the server spawns uses the same stack canary value

Imagine a server process that implements the function badfunc() (as below) which operates on input provided by the attacker. However, if the stack canary check ever fails, even once, the system will crash and the sysadmin will be alerted (ending the attacker's attempt to compromise the server). The attacker's goal is to divert control flow to shell code they have provided in input.

Assumptions: 32bit machine, No ASLR, No DEP, random Stack Canaries in use, but does not change across calls to badfunc(). The value of %esp when badfunc() is called is known to the attacker. The attacker can arrange to call badfunc() multiple times.

```
int badfunc(char *s) {
   char *ptr;
   char buf[64];

   ptr = s;

   strcpy(buf, ptr);
   printf("%x",(char)*ptr);

}
```

9. [4pts] Which of the following would be effective as part of an attack? all
   [✓ (a)] They could use the format string to read the value of the canary
   [✓ (b)] They could use pointer subterfuge to read the value of the canary
   (c) They could use pointer subterfuge to write the control data and skip the canary
   (d) They could overwrite the canary with a NULL so it any string comparison of it will exit out early.

10. [2pts] Assuming the ultimate goal is divert control to shell code that the attacker puts on the stack, how many times does badfunc need to be called before this can be achieved? one
   (a) one
   [✓ (b)] two
   (c) four
   (d) five
   (e) 1025 (on average)

Name: Nasser Almusser Student ID: A18507030

11. [4pts] The security group at UCSD has done a bunch of work looking at automotive security, including finding and remotely exploiting vulnerabilities in real cars. One of these actual vulnerabilities was with a popular CD player (if you're too young to have seen a CD player, they stored digital music on an optical disk, with each song on a different "track"). They discovered a vulnerability whereby a particularly formatted CD could take control of the player and ultimately the entire vehicle. Consider the following vulnerable C code (abstracted from this real-life example):

```
/* Information about the current CD. */

struct cd {
      unsigned int numtracks;      /* The number of tracks on this disc. */
      unsigned int tracklen[18];   /* The length of each track, in seconds. */
      void (*notify)(struct cd *);   /* Call this whenever the CD info changes. */
   };

struct cd *curcd = makestructcd();

/* Update the length of track number 'track'. */
   void update_cdinfo(unsigned int track, unsigned int newtracklen) {
      if (track > 18)
         return;
      curcd->tracklen[track] = newtracklen;
      (curcd->notify)(curcd);
   }
```

Don't worry about makestructcd(), it just allocates and initializes a struct cd on the heap. Assume the adversary can arrange for update_cdinfo() to be called with whatever values of track and newtracklen she likes (those values may have been read directly off the CD, for instance). Integers and pointers are 32bits long and data is allocated on the stack (and heap) in the order it is declared with no padding between fields. What values of *track* and *newtracklen* must the attacker choose to cause the program to jump to the shellcode they have previously stored at address 0x8048fa0? **(select one)**

(a) track = 0x8048fa0, newtracklen = 0
(b) track = 19, newtracklen = 0x8048fa0
[✓ (c)] track = 18, newtracklen = 0x8048fa0
(d) track = 0x8048f90, newtracklen=18
(e) track = 18, newtracklen = 0
(f) track = 18, newtracklen = 0x8048f90

12. [4pts] Continuing the previous attack, which of the following is true about the following mitigations that we discussed in class: **(select all that apply)**
(a) This attack could be stopped by stack canaries/cookies
(b) This attack could be stopped by DEP
(c) This attack could be stopped by control flow integrity
[✓ (d)] This attack could not be stopped by any of the above defenses.