CS161 Summer 2025 Introduction to Computer Security Midterm

Name: _____

Student ID: _____

This exam is 110 minutes long. There are 7 questions of varying credit. (100 points total)

Question:	1	2	3	4	5	6	7	Total
Points:	0	14	23	17	18	18	10	100

For questions with **circular bubbles**, you may select only one choice.

- ○ Unselected option (Completely unfilled)
- ◎ Don't do this (it will be graded as incorrect)
- ● Only one selected option (completely filled)

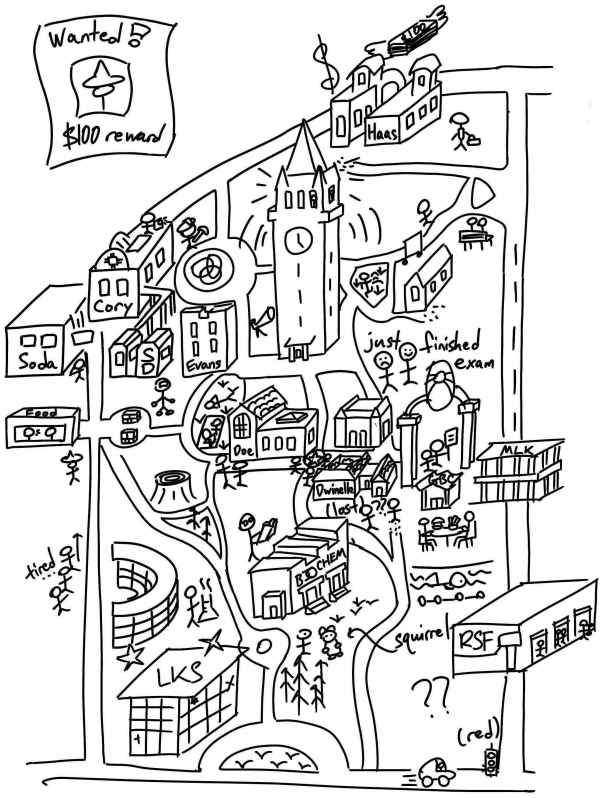
For questions with **square checkboxes**, you may select one or more choices.

- ■ You can select
- ■ multiple squares (completely filled).
- ☑ (Don't do this)

Anything you write outside the answer boxes or you ~~cross out~~ will not be graded. If you write multiple answers, your answer is ambiguous, or the bubble/checkbox is not entirely filled in, we may grade the worst interpretation.

Pre-Exam Activity (0 points):

EvanBot does not want to take their CS 161 Exam, so they are hiding somewhere on campus! Can you find them in time for the exam to start?



Artwork by Justin Yang ('28)

Q1 *Honor Code* **(0 points)**

I understand that I may not collaborate with anyone else on this exam, or cheat in any way. I am aware of the Berkeley Campus Code of Student Conduct and acknowledge that academic misconduct will be reported to the Center for Student Conduct and may further result in, at minimum, negative points on the exam.

Read the honor code above and sign your name: _____

Q2 *Potpourri* **(14 points)**

Each true/false is worth 1 point.

Q2.1 EvanBot purchases a $100,000 safe to protect a $100 necklace.

TRUE OR FALSE: The relevant security principle is Security is Economics.

◯ TRUE ◯ FALSE

Q2.2 EvanBot installs lasers in their office to further protect their safe.

TRUE OR FALSE: The relevant security principle is Defense in Depth.

◯ TRUE ◯ FALSE

Q2.3 EvanBot has no locks on their office, but they have cameras to detect intruders.

TRUE OR FALSE: The relevant security principle is Least Privilege.

◯ TRUE ◯ FALSE

Q2.4 TRUE OR FALSE: According to x86 calling convention, at the instant before the callee executes its first instruction, which values have already been pushed onto the stack?

◯ The RIP and arguments of the callee function.

◯ The RIP, SFP, and arguments of the callee function.

◯ The RIP of the callee function.

◯ A pointer to the middle of the callee's code.

Q2.5 TRUE OR FALSE: When memory is allocated for a struct in the heap, the first variable defined in the struct is stored at the lowest memory address.

◯ TRUE ◯ FALSE

Q2.6 TRUE OR FALSE: In big-endian format, the byte `0xde` of the 4-byte word `0xdeadbeef` is stored at the lowest memory address.

◯ TRUE ◯ FALSE

Q2.7 TRUE OR FALSE: The RIP of a `printf` function is located at `0xffffdb0c`. To process the first format specifier in the format string, the `printf` function consumes the argument at `0xffffdb14`.

◯ TRUE ◯ FALSE

Q2.8 TRUE OR FALSE: With pointer authentication codes enabled, overwriting only the least significant byte of the RIP will cause the program to crash.

◯ TRUE ◯ FALSE

Q2.9 True or False: If you need high performance, CTR mode is arguably better than CBC mode, because you can parallelize both encryption and decryption.

◯ True ◯ False

Q2.10 True or False: In CTR mode, if Mallory flips a bit in the ciphertext, the corresponding bit in the decrypted plaintext will also be flipped.

◯ True ◯ False

Q2.11 True or False: HMAC can be employed to create a secure, rollback-resistant PRNG.

◯ True ◯ False

Q2.12 True or False: NMAC requires exactly one symmetric key alongside the message as input.

◯ True ◯ False

Q2.13 True or False: Encrypt-then-MAC provides the same security properties as MAC-then-Encrypt.

◯ True ◯ False

Q2.14 True or False: A cryptographic hash function maps fixed-length inputs to arbitrary-length outputs.

◯ True ◯ False

Q3 *Memory Safety: Heap Calm and Carry On* 🇬🇧 **(23 points)**

Consider the following vulnerable C code:

```
1  typedef struct {
2      char tea[12];
3      char jam[4];
4  } crumpets;
5
6  typedef struct {
7      char milk[16];
8  } biscuit;
9
10 void UnionJack() {
11     int i = 0;
12     crumpets *c = malloc(sizeof(crumpets));
13     biscuit *b = malloc(sizeof(biscuit));
14     int max = 0x00000001;
15     void *target;
16     char buf[16];
17
18     while (i < max) {
19         if (i == 0) {
20             fgets(buf, 22, stdin);
21             memcpy(c->tea, buf, 16);
22         } else {
23             fread(buf, 16, 1, stdin);
24             memcpy(b->milk, buf, 16);
25         }
26         i++;
27     }
28     memcpy(target, &c, 4);
29 }
```

Stack at Line 16

RIP of `UnionJack`
SFP of `UnionJack`
`i`
`c`
(1)
`max`
(2)
(3)

Heap at Line 16

(4)
(5)
(6)

Assumptions:

- All memory safety defenses are disabled.
- `malloc` allocates memory starting at the lowest possible address with enough free space.
- `malloc` always allocates the exact amount of memory required by its input, with no metadata.
- No other process modifies the heap before or during this program's execution.
- The heap starts at address `0x0804b000` and grows upwards.
- You run GDB once and break at Line 16. You find that the RIP of `UnionJack` is located at `0xffffdc80`.
- Your goal is to place and execute a 32-byte `SHELLCODE`.

Q3.1 (1 point) Which of the following memory safety vulnerabilities are present in this code?

- ◯ Format string vulnerability
- ◯ Heap/stack overflow
- ◯ Signed/unsigned vulnerability
- ◯ None of the above

Q3.2 (1 point) What values go in blanks (1) through (3) in the stack diagram above?

- ◯ (1) `buf` (2) `b` (3) `c->jam`
- ◯ (1) `b` (2) `target` (3) `buf`
- ◯ (1) `b` (2) `b->milk` (3) `target`

Q3.3 (1 point) What values go in blanks (4) through (6) in the heap diagram above?

- ◯ (4) `b->milk` (5) `c->tea` (6) `c->jam`
- ◯ (4) `b->milk` (5) `c->jam` (6) `c->tea`
- ◯ (4) `c->jam` (5) `c->tea` (6) `b->milk`

Q3.4 (2 points) Which of these values does the exploit have to overwrite to execute `SHELLCODE`? Select all that apply.

- ☐ SFP of `UnionJack`
- ☐ RIP of `UnionJack`
- ☐ `target`
- ◯ None of the above

In the next two subparts, provide inputs that would cause the program to execute `SHELLCODE`. You may use slicing to construct payloads, e.g. `SHELLCODE[0:8]` represents the first 8 bytes of `SHELLCODE`.

Q3.5 (6 points) Input to `fgets` at Line 20:

Q3.6 (3 points) Input to `fread` at Line 23:

Q3.7 (2 points) Which memory safety defenses would cause the correct exploit (without modifications) to fail? Consider each choice independently.

- ☐ ASLR
- ☐ Stack canaries
- ◯ None of the above

Q3.8 (1 point) Would the correct exploit (without modifications) fail if non-executable pages are enabled?

- ◯ No, because the exploit redirects control flow to an executable heap region.
- ◯ No, because the exploit overwrites the return address without executing any injected code.
- ◯ Yes, because the injected shellcode is stored in a non-executable memory region.
- ◯ Yes, because the return address cannot be modified if non-executable pages are enabled.

Q3.9 (2 points) Would the correct exploit (without modifications) fail if Line 23 is replaced with `fgets(buf, 16, stdin)`?

- ◯ No, because `fgets` does not append null bytes, and therefore writes shellcode correctly.
- ◯ No, because both `fread` and `fgets` write exactly 16 bytes into `buf`.
- ◯ Yes, because `fgets` adds a null terminator, so only 15 bytes of user input will be read.
- ◯ Yes, because `fgets` reads from a different file stream than `fread`.

Q3.10 (4 points) Select the modifications that would prevent the attacker from executing `SHELLCODE`. Consider each choice independently.

- ☐ Changing Line 20 from `fgets(buf, 22, stdin);` to `fgets(buf, 17, stdin);`
- ☐ Changing Line 28 from `memcpy(target, &c, 4);` to `memcpy(target, c->tea, 4);`
- ☐ Changing Line 12 to `crumpets *c = malloc(sizeof(biscuit));` and Line 13 to `biscuit *b = malloc(sizeof(crumpets));`
- ☐ Moving Line 14 (`int max = 0x00000001;`) to be the first line of the function.
- ◯ None of the above

Q4 *Memory Safety: Are you being fr ?* (17 points)

Consider the following vulnerable C code:

```
1  void foo(void) {
2      char msg[8];
3      fgets(msg, 8, stdin);
4      printf(msg);
5      fread(msg, 20, 1, stdin);
6  }
7
8  void vulnerable() {
9      int i;
10     char buf[20];
11     char cpy[20];
12     fread(buf, 20, 1, stdin);
13     for(i = 19; i >= 0; i--) {
14         cpy[i] = buf[19-i];
15     }
16     foo();
17 }
```

Stack at Line 2

RIP of `vulnerable`
(1)
(2)
`i`
(3)
`cpy`
(4)
(5)
(6)
`msg`

Assumptions:

- Stack canaries and non-executable pages are enabled, and all other memory safety defenses are disabled.
- You run GDB once and find that the library function `system` is located at the address `0xdeadbeef`.
- The RIP of `foo` is located at `0xffffdc90`. The RIP of `vulnerable` is located at `0xffffdcc8`.
- Your goal is to execute `system` with the 8-character string `"rm -rf /"` as the argument.

Q4.1 (1 point) What values go in blanks (1) through (3) in the stack diagram above?

- ◯ (1) canary (2) RIP of `foo` (3) SFP of `vulnerable`
- ◯ (1) canary (2) SFP of `vulnerable` (3) `buf`
- ◯ (1) SFP of `vulnerable` (2) canary (3) `buf`
- ◯ (1) SFP of `vulnerable` (2) `cpy` (3) canary

Q4.2 (1 point) What values go in blanks (4) through (6) in the stack diagram above?

- ◯ (4) canary (5) RIP of `foo` (6) SFP of `foo`
- ◯ (4) canary (5) SFP of `foo` (6) RIP of `foo`
- ◯ (4) RIP of `foo` (5) SFP of `foo` (6) canary
- ◯ (4) RIP of `foo` (5) `&msg` (6) SFP of `foo`

Q4.3 (2 points) What type of vulnerability is present in this code? Select all that apply.

- ☐ Format string vulnerability
- ☐ Off-by-one
- ☐ ret2ret
- ☐ Buffer overflow
- ☐ Signed/unsigned
- ◯ None of the above

Q4.4 (3 points) Which of these inputs to `fgets` on Line 3 will always leak the value of the stack canary in the `foo` stack frame? Select all that apply.

Note: You are able to convert any numerical representation of the canary into a usable form.
Note: Stack canaries are four random bytes (no null byte).

- ☐ `'%x' * 3`
- ☐ `2 * '%c' + '%p'`
- ☐ `'%n' * 3`
- ☐ `'\xc0\xdc\xff\xff' + '%s'`
- ☐ `'\x88\xdc\xff\xff' + '%s'`
- ◯ None of the above

In the next two subparts, provide inputs that would cause the program to execute `system("rm -rf /")`. You may use `CANARY` to refer to the correct value of the stack canary, as leaked by `printf`.

Q4.5 (5 points) Input to `fread` at Line 12:

```
'A'*[          ] + '\x00' + '[                                ]' +
'[                                ]' + 'A'*[          ]
```

Q4.6 (5 points) Input to `fread` at Line 5:

Q5 *Cryptography: AES-161* 🤖 **(18 points)**

EvanBot creates a new block cipher mode of operation, called AES-161. The encryption formulas are:

$$C_1 = E_K(P_1 \oplus IV_1 \oplus IV_2)$$
$$C_n = E_K(P_n \oplus \underbrace{C_{n-1} \oplus ... \oplus C_1}_{\text{previous ciphertext blocks}} \oplus IV_1 \oplus IV_2)$$

In this entire question, assume that all IVs are independently randomly generated.

Q5.1 (2 points) Select the decryption formula for AES-161.

- ◯ $P_1 = D_K(C_1) \oplus IV_1 \oplus IV_2 \qquad P_n = D_K(C_n) \oplus C_{n-1} \oplus ... \oplus C_1 \oplus IV_1 \oplus IV_2$
- ◯ $P_1 = D_K(C_1) \oplus IV_1 \oplus IV_2 \qquad P_n = D_K(C_n) \oplus C_{n-1} \oplus ... \oplus C_1$
- ◯ $P_1 = D_K(C_1) \oplus IV_1 \oplus IV_2 \qquad P_n = D_K(C_{n-1}) \oplus ... \oplus D_K(C_1) \oplus IV_1 \oplus IV_2$
- ◯ $P_1 = D_K(C_1) \oplus IV_1 \oplus IV_2 \qquad P_n = D_K(C_n) \oplus C_{n-1} \oplus ... \oplus C_2 \oplus IV_1 \oplus IV_2$
- ◯ None of the above

Q5.2 (2 points) Is this scheme confidential under IND-CPA?

◯ Yes ◯ No

If "No," provide two plaintexts (M and M'), each two blocks long, that could be used by the adversary to win the IND-CPA game. You may write one integer per box, and they will be converted to the associated bitstrings.

If "Yes," leave the boxes blank.

$M = \left(\square , \square \right) \qquad M' = \left(\square , \square \right)$

Q5.3 (2 points) Alice sends a 4-block message (P_1, P_2, P_3, P_4) to Bob. Mallory tampers with the message by flipping one bit in C_3.

When Bob decrypts the tampered ciphertext, he gets (P_1', P_2', P_3', P_4'). Which blocks match the original plaintext that Alice sent? Select all that apply.

☐ P_1' ☐ P_2' ☐ P_3' ☐ P_4'

Q5.4 (1 point) Under this scheme, are encryption and decryption parallelizable?

- ◯ Both are parallelizable.
- ◯ Only decryption is parallelizable.
- ◯ Only encryption is parallelizable.
- ◯ Neither are parallelizable.

After looking at AES-161, EvanBot thinks that they have come up with a better idea. **For the following three subparts,** answer the same questions for this modified scheme:

$$C_1 = E_K(P_1 \oplus IV_1 \oplus IV_2)$$
$$C_n = E_K(P_n \oplus \underbrace{C_{n-1} \oplus ... \oplus C_1}_{\text{previous ciphertext blocks}} \oplus \underbrace{P_{n-1} \oplus ... \oplus P_1}_{\text{previous plaintext blocks}} \oplus IV_1 \oplus IV_2)$$

Q5.5 (2 points) Is this scheme confidential under IND-CPA?

◯ Yes ◯ No

If "No," provide two plaintexts (M and M'), each two blocks long, that could be used by the adversary to win the IND-CPA game. You may write one integer per box, and they will be converted to the associated bitstrings.

If "Yes," leave the boxes blank.

$M = \left(\boxed{}, \boxed{} \right) \qquad M' = \left(\boxed{}, \boxed{} \right)$

Q5.6 (2 points) Alice sends a 4-block message (P_1, P_2, P_3, P_4) to Bob. Mallory tampers with the message by flipping one bit in C_3.

When Bob decrypts the tampered ciphertext, he gets (P_1', P_2', P_3', P_4'). Which blocks match the original plaintext that Alice sent? Select all that apply.

☐ P_1' ☐ P_2' ☐ P_3' ☐ P_4'

Q5.7 (1 point) Are encryption and decryption under this scheme parallelizable?

◯ Both are parallelizable.
◯ Only decryption is parallelizable.
◯ Only encryption is parallelizable.
◯ Neither are parallelizable.

EvanBot wants to give scheming one last shot, so they make one last change. **For the following three subparts,** answer the same questions for this modified scheme:

$$C_1 = E_K(P_1 \oplus IV_1 \oplus IV_2)$$
$$C_n = E_K(P_n \oplus \underbrace{P_{n-1} \oplus ... \oplus P_1}_{\text{previous plaintext blocks}} \oplus IV_1 \oplus IV_2)$$

Q5.8 (2 points) Is this scheme confidential under IND-CPA?

◯ Yes ◯ No

If "No," provide two plaintexts (M and M'), each two blocks long, that could be used by the adversary to win the IND-CPA game. You may write one integer per box, and they will be converted to the associated bitstrings.

If "Yes," leave the boxes blank.

$M = \left(\boxed{}, \boxed{} \right) \qquad M' = \left(\boxed{}, \boxed{} \right)$

Q5.9 (2 points) Alice sends a 4-block message (P_1, P_2, P_3, P_4) to Bob. Mallory tampers with the message by flipping one bit in C_3.

When Bob decrypts the tampered ciphertext, he gets (P_1', P_2', P_3', P_4'). Which blocks match the original plaintext that Alice sent? Select all that apply.

☐ P_1' ☐ P_2' ☐ P_3' ☐ P_4'

Q5.10 (1 point) Are encryption and decryption under this scheme parallelizable?

◯ Both are parallelizable. ◯ Only decryption is parallelizable.

◯ Only encryption is parallelizable. ◯ Neither are parallelizable.

Q5.11 (1 point) If EvanBot decides to remove one of the IVs, does the confidentiality of this last scheme under IND-CPA change?

◯ Yes ◯ No

Q6 *Cryptography: Talk To Me Nicely* 🗣️ **(18 points)**

Alice and Bob are considering some cryptographic schemes. Determine whether each scheme provides confidentiality and integrity.

Notation:
- $M = M_1 \parallel M_2 \parallel ... \parallel M_n$ is the message.
- C is the resulting output sent over the channel.
- $K_1$and K_2 are secret keys known only to Alice and Bob.
- Every call to CBC uses independently randomly-generated IVs.

Note: For all schemes, each C_i is computed with a separate call to CBC*, with a separate IV.*

For the next two subparts, consider the following scheme:

$$C_i = \mathsf{CBC}(K_1, M_i) \qquad t_i = \mathsf{H}(C_i) \qquad C = (C_1 \parallel t_1) \parallel (C_2 \parallel t_2) \parallel ... \parallel (C_n \parallel t_n)$$

Q6.1 (1 point) Does this scheme provide confidentiality?

◯ Yes ◯ No

Q6.2 (4 points) Does this scheme provide integrity?

◯ Yes ◯ No

Explain your reasoning for Q6.2:

For the next two subparts, consider the following scheme:

$$C_i = \mathsf{CBC}(K_1, M_i) \qquad t_i = \mathsf{HMAC}(K_2, C_i) \qquad C = (C_1 \parallel t_1) \parallel (C_2 \parallel t_2) \parallel ... \parallel (C_n \parallel t_n)$$

Q6.3 (1 point) Does this scheme provide confidentiality?

◯ Yes ◯ No

Q6.4 (4 points) Does this scheme provide integrity?

◯ Yes ◯ No

Explain your reasoning for Q6.4:

For the next two subparts, consider the following scheme where $C_0 = \text{IV}$:

$$C_i = \mathsf{CBC}(K_1, M_i) \qquad t_i = H(C_i \oplus C_{i-1}) \qquad C = (C_1 \parallel t_1) \parallel (C_2 \parallel t_2) \parallel ... \parallel (C_n \parallel t_n)$$

Q6.5 (1 point) Does this scheme provide confidentiality?

◯ Yes ◯ No

Q6.6 (4 points) Does this scheme provide integrity?

◯ Yes ◯ No

Explain your reasoning for Q6.6:

These last two subparts are **independent from earlier subparts.**

Q6.7 (2 points) EvanBot uses a hash function with a 256-bit output. Approximately how many random inputs would EvanBot need to hash before expecting to find a collision?

◯ 2^{32} ◯ 2^{256}

◯ 2^{128} ◯ 2^{512}

◯ 2^{192} ◯ None of the above

Q6.8 (1 point) Select all properties of a cryptographic hash function.

☐ Deterministic ☐ Collision resistance

☐ Invertible ☐ Resistance to length-extension attacks

☐ One-wayness ◯ None of the above

Q7 *Cryptography: El-Ephant in the Room* 🐘 **(10 points)**

Alice and Bob design a protocol for communicating.

Before the start of the protocol:
- Alice and Bob agree on a large prime p, generator g, and password pwd.
- Bob has a private key b and a known public key $B \equiv g^b \bmod p$.

Each time they wish to communicate, they do these steps:

1. **Alice and Bob derive the password key** by each computing $K_{\text{pwd}} = \mathsf{H}(\text{pwd})$.
2. **Alice randomly generates a session key** K_{sess}.
3. Alice picks a random exponent u and uses El Gamal to encrypt K_{sess} with Bob's public key B:

$$U \equiv g^u \bmod p$$
$$V \equiv K_{\text{sess}} \cdot B^u \bmod p$$

   Alice then computes $C = \mathsf{Enc}\big(K_{\text{pwd}}, (U \parallel V)\big)$, and **Alice sends C to Bob.**
4. **Bob recovers the session key** by first using K_{pwd} to decrypt C to get $(U \parallel V)$.

   Then, he uses his private key b to compute $K_{\text{sess}} =$ _____ (Q7.1).

Q7.1 (2 points) Which equation describes how Bob computes the session key in Step 4?

- ◯ $V \cdot \left(U^b\right)^{-1} \bmod p$
- ◯ $\left(V \cdot U^{-1}\right)^b \bmod p$
- ◯ $U^b \cdot V^{-1} \bmod p$
- ◯ $V^b \cdot U \bmod p$

Eve is an attacker who records $C = \mathsf{Enc}\big(K_{\text{pwd}}, (U \parallel V)\big)$.

Q7.2 (2 points) What is the minimum set of values Eve needs to derive K_{sess}?

- ◯ Both b and pwd.
- ◯ pwd (but not b).
- ◯ b (but not pwd).
- ◯ Neither b nor pwd.

Q7.3 (2 points) For this subpart only, suppose Eve knows pwd and b. Can Eve now derive u?

In 10 words or fewer, explain your reasoning. (The staff answer is 3 words.)

◯ Yes ◯ No

Q7.4 (2 points) Step 3 uses K_{pwd} to encrypt $(U \parallel V)$. Select all encryption schemes for Step 3 that would (with high probability) prevent Alice and Bob from computing the same value of C twice.

- ☐ AES-ECB
- ☐ AES-CBC with random IVs
- ☐ AES-CBC with IVs always set to 0
- ☐ AES-CTR with random nonces
- ☐ AES-CTR with nonces always set to 0
- ◯ None of the above

Q7.5 (2 points) For this subpart only, we modify the protocol so that $U \parallel V$ is no longer encrypted, but an HMAC is applied instead:

In Step 3, Alice now computes $C = U \parallel V \parallel \mathsf{HMAC}\big(K_{\text{pwd}}, (U \parallel V)\big)$.

Suppose Eve knows b. Can Eve derive K_{sess}?

- ◯ Yes, but only if Eve knows pwd.
- ◯ Yes, even if Eve does not know pwd.
- ◯ No, even if Eve knows pwd.

(Question 7 continued...)

Post-Exam Activity: Bot Gets Dinner

EvanBot is going out to get dinner after their CS 161 exam! Where does Bot want to eat?

Comment Box

Congratulations for making it to the end of the exam! Feel free to leave any final thoughts, comments, feedback, or doodles here: