

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	9	16	22	15	13	25	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 ~~eross 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)



According to the official CS 161 lore, what is Alice's day job?

- ☐ Professor
- ☐ Student
- ☐ Mailman
- ☐ Architect
- ☐ Spy
- ☐ Other: _____

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: A Little Bit of Everything 🍷

(9 points)

Each question is worth 1 point.

Q2.1 Consider a C compiler that leaves stack canaries off unless the programmer uses a special flag.

TRUE OR FALSE: This compiler is correctly following the “use fail-safe defaults” security principle.

☐ TRUE ☐ FALSE

Q2.2 TRUE OR FALSE: In the struct shown to the right, `p.a` lives at a lower memory address than `p.b`.

```
struct {  
    int a;  
    int b;  
} p;
```

☐ TRUE ☐ FALSE

Q2.3 A 32-bit program uses stack canaries whose first byte is always a null byte, as described in lecture. How many unique values of the canary are there?

☐ 2^8 ☐ 2^{24} ☐ 2^{32} ☐ 2^{56}

Q2.4 TRUE OR FALSE: Pointer authentication is secure against an attacker trying to brute-force the value of the PAC (pointer authentication code).

☐ TRUE ☐ FALSE

Q2.5 TRUE OR FALSE: To use a NOP sled, an attacker should overwrite the RIP/RA of a function with no-op instructions (e.g. write `0x90909090` at the RIP/RA).

☐ TRUE ☐ FALSE

Q2.6 Alice sends $M, H(M)$ to Bob over a channel that Mallory controls, where H is SHA-256. Bob recomputes $H(M)$ and accepts M if the hash he computed matches the hash he received.

TRUE OR FALSE: This lets Bob detect any modification Mallory makes to M .

☐ TRUE ☐ FALSE

Q2.7 Bob uses Trust on First Use (TOFU) for Alice’s public key. Mallory tampers with Alice’s public key on Bob’s very first connection to Alice.

TRUE OR FALSE: On Bob’s later connections to Alice, Bob will flag Alice’s real public key as a mismatch.

☐ TRUE ☐ FALSE

Q2.8 TRUE OR FALSE: If Bob receives a certificate for `alice.com` that is correctly signed by a certificate authority (CA) he trusts, he can conclude that the party who sent him the certificate is `alice.com`.

☐ TRUE ☐ FALSE

Q2.9 TRUE OR FALSE: Decrypting an AES-CTR ciphertext runs AES only in its encode (encryption) direction, and never uses AES’s decode (decryption) function.

☐ TRUE ☐ FALSE

Q3 Memory Safety: Negative Attention

(16 points)

Consider the following vulnerable C code:

```

1 void set_score(int *scores, int idx, int val) {
2     if (idx > 7) {
3         return;
4     }
5     scores[idx] = val;
6 }
7
8 void game(int *packet) {
9     int scores[8];
10    set_score(scores, packet[0], packet[1]);
11 }

```

Stack at Line 5

RIP/RA of game
SFP of game
scores
(1)
(2)
(3)
RIP/RA of set_score
SFP of set_score

Assumptions:

- All memory-safety defenses are disabled.
- You run GDB once and find that **scores** starts at address **0xffffd6c0**.
- **packet** points to an integer array, and you control the values in this array.
- Your goal is to execute shellcode that is already placed in memory. You run GDB once and find that the shellcode starts at address **0xdeadbeef**.

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

Q3.1 Blank (1): ☐ **val** ☐ **idx** ☐ **scores**

Q3.2 Blank (2): ☐ **val** ☐ **idx** ☐ **scores**

Q3.3 Blank (3): ☐ **val** ☐ **idx** ☐ **scores**

Q3.4 (8 points) Provide inputs to the **packet** array that would cause the program to execute shellcode.

In each box, write a single integer in decimal (e.g. -71) or hexadecimal (e.g. 0x12345678).

You must write 0x in front of any hexadecimal values. Do not worry about little-endianness.

packet [0] = **packet** [1] =

Q3.5 (3 points) Which of these defenses would cause the correct exploit (without modification) to fail? Consider each defense independently. Select all that apply.

- ☐ Non-executable pages (with shellcode on a non-executable page) ☐ ASLR
- ☐ Replace Line 2 with `if (idx < 0 || idx > 7)` ☐ None of the above

Q3.6 (2 points) For this subpart only, suppose stack canaries are enabled. Why can the correct exploit (possibly with modifications) still succeed with stack canaries enabled?

- ☐ The exploit reads the canary, and overwrites the canary with itself.
- ☐ The exploit does not modify the canary, so the canary check passes.
- ☐ The exploit jumps to shellcode before the canary is ever checked.

Q4 Memory Safety: Null and Void

(22 points)

Consider the following vulnerable C code:

```

1 void copy_record(void) {
2     char name[64];
3     int len = fread(name, 1, 64, stdin);
4     name[len] = '\0';
5 }
6
7 void vulnerable() {
8     copy_record();
9 }

```

Stack at Line 4

RIP/RA of vulnerable
SFP of vulnerable
(1)
(2)
name (64 bytes)
(3)

Assumptions:

- All memory-safety defenses are disabled.
- You run GDB once and find that:
 - **name** starts at address **0xffffd700**.
 - The SFP of **copy_record** has the **value** **0xffffd748**.
- **fread** returns the number of bytes read.
- Your goal is to place and execute a 44-byte long **SHELLCODE**.

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

Q4.1 Blank (1): ☐ RIP/RA of **copy_record** ☐ SFP of **copy_record** ☐ **len**

Q4.2 Blank (2): ☐ RIP/RA of **copy_record** ☐ SFP of **copy_record** ☐ **len**

Q4.3 Blank (3): ☐ RIP/RA of **copy_record** ☐ SFP of **copy_record** ☐ **len**

Q4.4 (2 points) If you input exactly 64 bytes into **fread**, Line 4 writes one byte past the end of **name**.

What is the **value** of the SFP of **copy_record** immediately after Line 4 executes?

☐ 0xffffd600 ☐ 0xffffd700 ☐ 0xffffd740 ☐ 0xffffd748

Q4.5 (8 points) Provide an input to the **fread** on Line 3 that causes the program to execute **SHELLCODE**.

If a part of the input can be any non-zero value, use '**A**' * **n** to represent **n** bytes of garbage.

(Question 4 continued...)

Q4.6 (3 points) When does the correct exploit start executing instructions in **SHELLCODE**?

- ☐ Immediately after the **fread** on Line 3 returns.
- ☐ Immediately after **name[len] = '\0'** on Line 4 executes.
- ☐ Immediately after **copy_record** returns.
- ☐ Immediately after **vulnerable** returns.

Q4.7 (3 points) Which of these defenses would cause the correct exploit (without modification) to fail? Consider each defense independently. Select all that apply.

- | | |
|---|---|
| ☐ Non-executable pages | ☐ ASLR |
| ☐ Stack canaries | ☐ None of the above |

Q4.8 (3 points) Reminder: all defenses are still disabled.

Suppose that you run the same code on a different machine.
(Assume the program does not run any other code before the provided code.)

On this machine, you find that:

- **name** starts at address **0xffffd780**.
- The SFP of **copy_record** has the **value** **0xffffd7c8**.

As before, Line 4 modifies the value of the SFP of **copy_record**.

Is it possible to execute an off-by-one attack on this machine?

- ☐ Yes, because the modified SFP and the address of **name** have the same second-to-last byte.
- ☐ Yes, because the modified SFP is an address that points inside the **name** array.
- ☐ No, because the modified SFP and the address of **name** have different second-to-last bytes.
- ☐ No, because the modified SFP is an address that points far outside the **name** array.

Q5 Authentication: All You Can Answer!**(15 points)**

Each subpart is independent.

Q5.1 (2 points) Let F be a secure pseudorandom function (PRF) on fixed-length inputs, and let H be a collision-resistant hash function with output length equal to F 's input length. Alice and Bob share a random key K .

Which of these is an EUF-CMA secure MAC for arbitrary-length messages M ?

- ☐ $H(M)$ ☐ $K \oplus H(M)$ ☐ $F(K, H(M))$ ☐ $H(K) \oplus H(M)$

Q5.2 (2 points) In the RSA signature scheme from lecture, Alice's verification key is (N, e) , her signing key is d , and she signs M as $\sigma = H(M)^d \bmod N$.

Bob verifies (M, σ) by checking if the following condition holds:

- ☐ $H(M) = \sigma^e \bmod N$ ☐ $\sigma = H(M)^d \bmod N$
☐ $\sigma = H(M)^e \bmod N$ ☐ $H(M)^e = \sigma^d \bmod N$

Q5.3 (5 points) Alice encrypts two equal-length messages M_1 and M_2 with AES-CTR under the same key K , but uses the same nonce r both times.

Eve sees both ciphertexts C_1 and C_2 (and r) but does not know K . Select all pieces of information that Eve can learn.

- ☐ $M_1 \oplus M_2$ ☐ M_2 , given that she already knows M_1
☐ Whether $M_1 = M_2$ ☐ M_2 , without knowing M_1
☐ The key K ☐ None of the above

Q5.4 (2 points) Given an IND-CPA secure encryption scheme Enc , and an EUF-CMA secure MAC, which construction is the Encrypt-then-MAC authenticated encryption scheme recommended in lecture?

- ☐ $\text{MAC}(K_1, M) \parallel \text{Enc}(K_2, M)$. ☐ $C \parallel \text{MAC}(K_1, C)$, where $C = \text{Enc}(K_2, M)$.
☐ $\text{Enc}(K_2, M \parallel \text{MAC}(K_1, M))$. ☐ $C \parallel \text{MAC}(K, C)$, where $C = \text{Enc}(K, M)$.

Q5.5 (4 points) Alice and Bob run ephemeral Diffie-Hellman to compute a shared key K . Select all true statements.

- ☐ An eavesdropper can compute K as $g^a + g^b \bmod p$.
☐ An eavesdropper can compute K as $g^a \cdot g^b \bmod p$.
☐ The exchange by itself lets Alice confirm that the party she derived K with is really Bob.
☐ If Alice and Bob generate a fresh a, b for each session and then discard a, b , and K , an attacker who later breaks into both machines cannot decrypt previously recorded sessions.
☐ None of the above

Q6 Authentication: Chain of Command

(13 points)

EvanBot wants to learn the public key of `shop.oski.edu`.

EvanBot trusts the public verification key VK_{Root} as a trust anchor.

`shop.oski.edu` presents the following two certificates to EvanBot:

1. Signed by SK_{Root} : “CA has public key VK_{CA} , and may issue certificates for `*.oski.edu`.”
2. Signed by SK_{CA} : “`shop.oski.edu` has public key VK_{Oski} , and may issue certificates for no one.”

Q6.1 (2 points) What value should EvanBot use to verify the signature in Certificate 2?

- ☐ The trust anchor VK_{Root} .
- ☐ The value VK_{CA} from Certificate 1.
- ☐ The value VK_{Oski} from Certificate 2.

Q6.2 (2 points) EvanBot sees the following certificate:

Signed by SK_{M} : “`dev.oski.edu` has public key VK_{M} , and may issue certificates for no one.”

The signature in this certificate is valid. Can EvanBot trust VK_{M} ?

- ☐ Yes, because EvanBot can use VK_{M} in this certificate to verify the signature.
- ☐ Yes, because this certificate indicates that CA is endorsing VK_{M} .
- ☐ No, because this certificate must be signed by either VK_{Root} or VK_{CA} to be trusted.
- ☐ No, because this certificate must be signed by VK_{Root} to be trusted.

Q6.3 (3 points) In this subpart, Mallory is an attacker who knows SK_{CA} (but not SK_{Root} or SK_{Oski}).

Mallory’s own key pair is $(VK_{\text{Mallory}}, SK_{\text{Mallory}})$.

Select all attacks that Mallory can carry out.

- ☐ Mallory can forge a certificate binding `bank.oski.edu` to VK_{Mallory} that EvanBot accepts.
- ☐ Mallory can forge a certificate binding `www.google.com` to VK_{Mallory} that EvanBot accepts.
- ☐ Mallory can impersonate `shop.oski.edu` to EvanBot without knowing SK_{Oski} .
- ☐ None of the above

(Question 6 continued...)

The text on the previous page, reprinted for your convenience:

EvanBot trusts the public verification key VK_{Root} as a trust anchor.

shop.oski.edu presents the following two certificates to EvanBot:

1. Signed by SK_{Root} : “CA has public key VK_{CA} , and may issue certificates for ***.oski.edu**.”
2. Signed by SK_{CA} : “**shop.oski.edu** has public key VK_{Oski} , and may issue certificates for no one.”

Q6.4 (2 points) In this subpart, suppose Mallory knows SK_{CA} (but not SK_{Root} or SK_{Oski}).

True or false: If we implement a short expiration time on all certificates, then Mallory will become harmless after a short time.

☐ True ☐ False

Q6.5 (2 points) In this subpart, suppose Mallory knows SK_{Root} (but not SK_{CA} or SK_{Oski}).

True or false: If we implement a short expiration time on all certificates, then Mallory will become harmless after a short time.

☐ True ☐ False

Q6.6 (2 points) Suppose EvanBot checks for bad certificates by periodically downloading a certificate revocation list (CRL).

Which of these best describes the trade-off between security and performance of using this approach, compared to doing nothing?

- | | |
|--|---|
| ☐ Better security, better performance. | ☐ Worse security, better performance. |
| ☐ Better security, worse performance. | ☐ Worse security, worse performance. |

(25 points)

Recall the ElGamal encryption scheme from lecture:

- The public parameters are p and g .
- Bob's private key is b , and his public key is $B = g^b \bmod p$.
- To encrypt a message m , the sender picks a fresh random exponent r and computes the ciphertext

$$(c_1, c_2) = (g^r \bmod p, m \cdot B^r \bmod p).$$

- To decrypt, Bob computes $m = c_2 \cdot (c_1^b)^{-1} \bmod p$.

Q7.1 (5 points) For this subpart only, use the following values:

$$p = 23, \quad g = 5, \quad b = 6, \quad B = 5^6 \bmod 23 = 8.$$

Alice encrypts a message m and sends ciphertext $(c_1, c_2) = (19, 7)$ to Bob.

Mallory does not know a or m , but she wants to change the ciphertext to (c'_1, c'_2) , so that when Bob decrypts, Bob receives the message $3m \bmod 23$. What should the tampered ciphertext be?

Write one integer between 0 and 22 in each box.

$$c'_1 = \boxed{} \qquad c'_2 = \boxed{}$$

Q7.2 (2 points) Alice encrypts two messages:

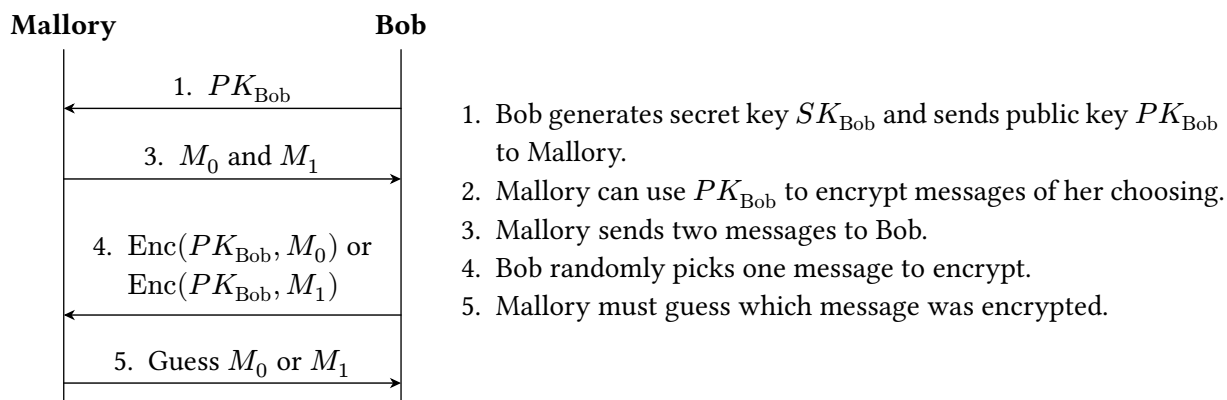
- She encrypts m_1 and sends ciphertext (c_1, c_2) to Bob.
- She encrypts m_2 and sends ciphertext (d_1, d_2) to Bob.

Mallory intercepts both ciphertexts. Which tampered ciphertext decrypts to $m_1 \cdot m_2 \bmod p$?

- $$\begin{array}{ll} \bigcirc (c_1 \cdot d_1 \bmod p, c_2 \cdot d_2 \bmod p) & \bigcirc (c_1 + d_1 \bmod p, c_2 + d_2 \bmod p) \\ \bigcirc (c_1 \cdot d_2 \bmod p, c_2 \cdot d_1 \bmod p) & \bigcirc (c_1^{d_1} \bmod p, c_2^{d_2} \bmod p) \end{array}$$

(Question 7 continued...)

Recall the IND-CPA game for public-key encryption schemes (from lecture):



Q7.3 (4 points) Mallory tries several strategies to win the IND-CPA game against ElGamal.

For all of her strategies in this subpart:

- In Step 2: Mallory uses PK_{Bob} to compute the encryptions of two messages x and $2x \bmod p$.
- In Step 3: Mallory sends $M_0 = x$ and $M_1 = 2x \bmod p$ to Bob.
- In Step 5: *see answer choices*

Which strategies allow Mallory to win the IND-CPA game with probability $> 0.5 + \text{negligible}$?
Select all that apply.

- ☐ Use Bob's encryption in Step 4 to compute the r he used. Then, use r to compute the shared secret $B^r = g^{br} \bmod p$ and decrypt the message that Bob chose.
- ☐ If Bob's encryption in Step 4 matches Mallory's encryption of x in Step 2, guess M_0 .
Else, guess M_1 .
- ☐ If Bob returns a value $\geq p/2$, guess that $M_1 = 2x \bmod p$ was encrypted.
Else, guess that $M_0 = x$ was encrypted.
- ☐ Flip a fair coin and guess M_0 and M_1 at random.
- ☐ None of the above

Q7.4 (4 points) Mallory tries another strategy to win the IND-CPA game against ElGamal:

- In Step 3: Mallory sends two randomly-generated messages M_0 and M_1 to Bob.
- In Step 5: Use Bob's encryption in Step 4 to compute the r he used. Then, use r to compute the shared secret $B^r = g^{br} \bmod p$ and decrypt the message that Bob chose.

Does this allow Mallory to win the IND-CPA game with probability $> 0.5 + \text{negligible}$?

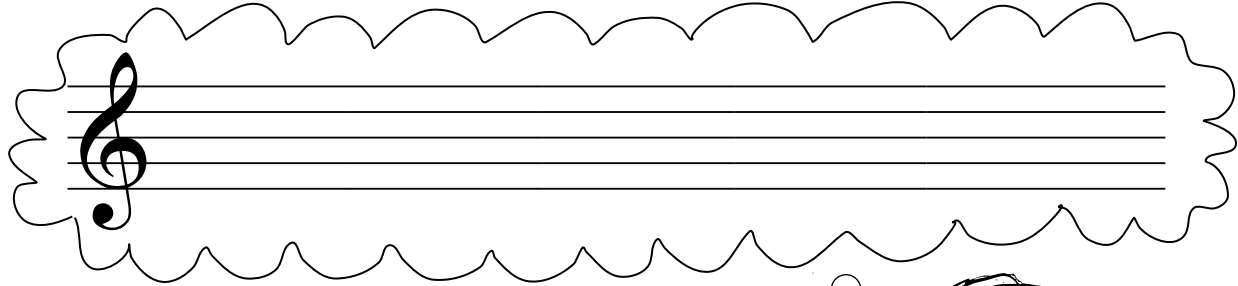
- ☐ Yes ☐ No

Explain why or why not in 20 words or fewer.

Nothing on this page will affect your grade.

Post-Exam Activity: Earworm

Got a song stuck in your head? If you let Eve eavesdrop a few bars, maybe she can figure out what it is!



Comment Box

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