Madison & Ana
Summer 2023

CS 161 Computer Security Midterm

PRINT your name: ______________________, ______________________
(last) (first)

PRINT your student ID: ________________________________

You have 120 minutes. There are 7 questions of varying credit (150 points total).

Question:	1	2	3	4	5	6	7	Total
Points:	3	30	23	24	20	24	26	150

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

○ Unselected option (completely unfilled)

● 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)

Pre-exam activity (for fun, not graded):
Word search! Circle your favorite mascot(s).

Q	E	N	2	A	M
5	E	V	E	L	A
C	O	D	A	O	L
B	E	E	P	N	L
O	C	E	I	Z	O
S	I	R	N	O	R
K	L	T	T	V	Y
I	A	B	O	B	6

Q1 *Honor Code* (3 points)

Read the following honor code and sign your name.

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.

SIGN your name:

____

Q2 *True/False* **(30 points)**

Each true/false is worth 2 points.

Q2.1 True or False: Time of check to time of use (TOCTTOU) vulnerabilities violate the security principle "ensure complete mediation".

○ (A) True ○ (B) False

Q2.2 True or False: In the real world (outside of this class), stack canaries usually have a null byte.

○ (A) True ○ (B) False

Q2.3 True or False: It is often worth disabling stack canaries in order to save compiler overhead.

○ (A) True ○ (B) False

Q2.4 True or False: In practice, PACs are a useful defense on an x86 little-endian 16-bit system.

○ (A) True ○ (B) False

Q2.5 True or False: Faster hash algorithms are strictly better than slow hash algorithms.

○ (A) True ○ (B) False

Q2.6 True or False: Using the same key to both encrypt and sign a message is good practice because you need to remember fewer keys.

○ (A) True ○ (B) False

Q2.7 True or False: A passive eavesdropper is able to compromise the security properties of Diffie-Hellman.

○ (A) True ○ (B) False

Q2.8 True or False: El Gamal encryption is a malleable scheme.

○ (A) True ○ (B) False

Q2.9 Consider the following scheme: A message M is encrypted with AES-CBC (with a random IV). The ciphertext is then tagged with a MAC scheme that fails EU-CPA security. Then, both the ciphertext and the tag are outputted.

TRUE or FALSE: This scheme preserves the confidentiality of M.

○ (A) TRUE ○ (B) FALSE

Q2.10 TRUE or FALSE: IND-CPA secure schemes must be non-deterministic.

○ (A) TRUE ○ (B) FALSE

Q2.11 TRUE or FALSE: Once a certificate is issued, it is impossible to revoke it.

○ (A) TRUE ○ (B) FALSE

Q2.12 TRUE or FALSE: Certificate implementations using a trusted directory are scalable.

○ (A) TRUE ○ (B) FALSE

Q2.13 TRUE or FALSE: Storing passwords as $\mathsf{Enc}(K, \texttt{pwd})$ with secret key K is more secure than hashing them as $\mathsf{H}(\texttt{pwd})$

Clarification during exam: Instead of $\mathsf{H}(\texttt{pwd})$, passwords are stored as $\mathsf{H}(\texttt{pwd} \| \texttt{salt})$ where salt is a randomly chosen value for each user.

○ (A) TRUE ○ (B) FALSE

Q2.14 Evanbot tried to design a secure file system, but forgot to add integrity and authenticity features. He's now struggling to rewrite the entire codebase to add these features.

TRUE or FALSE: Evanbot violated Design in Security from the Start.

○ (A) TRUE ○ (B) FALSE

Q2.15 CS161 staff spent all semester implementing a single strong security feature to their new website. They figured this would be enough to prevent any pesky attackers, so they stored the exam on the website. Unfortunately, a student Mallory, was able to bypass the feature, and now has access to the exam!

TRUE or FALSE: CS161 staff violated Separation of Responsibility.

○ (A) TRUE ○ (B) FALSE

Q3 *Memory Safety Exploit: Across the Security-Verse* (23 points)

Consider the following code:

```
 1 void verse() {
 2     char miles[256];
 3     fgets(miles, 257, stdin);
 4 }
 5
 6 void spider() {
 7     verse();
 8 }
 9
10 void main() {
11     char *peter = (char *) malloc(128);
12     gets(peter);
13     printf("%x", peter);
14     spider();
15 }
```

Stack at Line 2
RIP of `main`
SFP of `main`
(1)
(2)
(3)
(4)
(5)
(6)

Assumptions:

- You may use `SHELLCODE` as a 120-byte shellcode.
- **ASLR is enabled** (including the code segment), but all other memory safety defenses are disabled.
- Assume that ASLR will **always** randomize the value of the SFP of `verse` to end with a least significant byte (LSB) in the range of `0x10` - `0xfc` (e.g. the LSB will never be `0x00` or `0x04`).
- You may use the variable `ADDR` to represent the output of `printf` on line 13, converted into a 4-byte, little-endian, byte string. You can directly use this without casting, converting, or slicing.

For the following 2 subparts, one variable should go in each row of the stack diagram. Assume the program is paused at a breakpoint on line 2.

Q3.1 (2 points) What values go in blanks (1), (2), and (3) in the stack diagram above?

- ○ (A) (1) RIP of `spider` (2) SFP of `spider` (3) `peter`
- ○ (B) (1) `peter` (2) RIP of `spider` (3) SFP of `spider`
- ○ (C) (1) RIP of `spider` (2) `peter` (3) SFP of `spider`

Q3.2 (2 points) What values go in blanks (4), (5), and (6) in the stack diagram above?

- ○ (A) (4) SFP of `verse` (5) RIP of `verse` (6) `peter`
- ○ (B) (4) RIP of `verse` (5) SFP of `verse` (6) `miles[256]`
- ○ (C) (4) `miles[256]` (5) RIP of `verse` (6) SFP of `verse`

Q3.3 (10 points) Provide inputs to the program that will execute shellcode with 100% probability. Use Python syntax (from project 1).

Input to `gets` on Line 12:

Input to `fgets` on Line 3:

Q3.4 (4 points) If we replaced line 13 with `printf("%x", &peter);`, is an exploit that executes `SHELLCODE` still possible?

- ○ (A) Yes, with 100% probability
- ○ (C) No
- ○ (B) Yes, with probability less than 100%

Q3.5 (5 points) Assume your exploit in Q3.3 was correct. **For this subpart only**, assume the LSB of the SFP of `verse` is `0x00`. Would this exploit still work?

- ○ (A) Yes
- ○ (B) No

Briefly justify your answer.

Q4 *Memory Safety Exploit: Snacktime* (24 points)

Consider the following code:

```
 1 void goldfish(char* potato) {
 2     fgets(potato, 256, stdin);
 3
 4     int8_t chip = strlen(potato);
 5     printf("%s", &potato[chip]);
 6
 7     gets(potato);
 8 }
 9
10 void main() {
11     char cola[256];
12     goldfish(cola);
13 }
```

Stack at Line 4

RIP of `main`
SFP of `main`
(1)
(2)
(3)
(4)
(5)
(6)
(7)

Assumptions:

- **Stack canaries are enabled**, and all other memory safety defenses are disabled (unless otherwise specified).
- None of the bytes of any stack canaries are a null byte.
- You run GDB once and find that the `cola` buffer is located at the address `0xff00caf0`.

For the following 2 subparts, assume the program is paused at a breakpoint on line 4.

Q4.1 (2 points) Which values go in blanks (1), (2), and (3) in the stack diagram above?

- ○ (A) (1) `cola[256]` (2) canary (3) `potato`
- ○ (B) (1) canary (2) `cola[256]` (3) RIP of `goldfish`
- ○ (C) (1) canary (2) `potato` (3) `cola[256]`
- ○ (D) (1) canary (2) `cola[256]` (3) `potato`

Q4.2 (2 points) Which values go in blanks (4), (5), (6), and (7) in the stack diagram above?

- ○ (A) (4) SFP of `goldfish` (5) canary (6) `chip` (7) `potato`
- ○ (B) (4) RIP of `goldfish` (5) SFP of `goldfish` (6) canary (7) `chip`
- ○ (C) (4) SFP of `goldfish` (5) `potato` (6) canary (7) `chip`
- ○ (D) (4) RIP of `goldfish` (5) SFP of `goldfish` (6) `chip` (7) canary

Q4.3 (3 points) Which vulnerability is present in the code?

- ○ (A) Off-by-one vulnerability
- ○ (B) Signed/unsigned vulnerability
- ○ (C) Format string vulnerability
- ○ (D) None of the above

Q4.4 (10 points) Assuming you have a 252-byte `SHELLCODE`, provide inputs for `fgets` and `gets` that executes the `SHELLCODE`. You may use the variable `OUTPUT` to represent the output of `printf` on line 5. Use Python syntax (from project 1).

Reminder: On one execution of the program, all stack frames have the same canary value.

`fgets`:

`gets`:

Q4.5 (2 points) **For this subpart only**, assume that non-executable pages are enabled (in addition to stack canaries).

True or False: The exploit from the previous subpart would still cause the shellcode to be executed.

○ (A) True ○ (B) False

Q4.6 (5 points) **For this subpart only**, assume that ASLR is enabled (in addition to stack canaries).

Would it still be possible to exploit this program with overwhelming probability?

○ (A) Possible ○ (B) Not Possible

If you selected *Possible*, what conditions need to be true for the exploit to succeed? If you selected *Not Possible*, list which addresses in your exploit from above could no longer be learned by the attacker.

Q5 *Cryptography: All or Nothing Security* **(20 points)**

EvanBot decides to modify AES-CTR in order to provide **all-or-nothing security**. All-or-nothing security means that modifying *any* part of the ciphertext will make the *entire* plaintext decrypt to some sort of "garbage" output.

EvanBot designs the following scheme to encrypt $M = (M_1, M_2, \ldots, M_n)$:

1. EvanBot generates a new random key K_2 on top of the original key K_1. Note that K_2 is **not** known to the decryptor, even though K_1 is.
2. EvanBot transforms M into a "pseudomessage" M' by setting $M'_i = M_i \oplus E_{K_2}(i)$.
3. EvanBot adds the block $M'_{n+1} = H(M'_1 \oplus 1) \oplus H(M'_2 \oplus 2) \oplus \ldots \oplus H(M'_n \oplus n) \oplus K_2$.
4. EvanBot derives the ciphertext $C = \mathsf{Enc}(K_1, M')$ using AES-CTR with key K_1 and IV IV.

First, we will walk through the decryption process for this all-or-nothing scheme. Fill in the blanks for the following by answering the multiple-choice subparts below:

1. CodaBot receives C.
2. CodaBot decrypts C with key K_1 to recover ________________.
3. CodaBot sets K_2 = $M'_{n+1} \oplus$ ________________.
4. CodaBot finds i-th original message block as $M_i =$ ________________.

Q5.1 (2 points) Select the correct option for the blank on Step 2:

○ (A) K_2

○ (B) $H(M'_1 \oplus 1) \oplus \ldots \oplus H(M'_n \oplus n)$

○ (C) $M'_i \oplus E_{K_2}(i)$

○ (D) M'

Q5.2 (2 points) Select the correct option for the blank on Step 3:

○ (A) K_2

○ (B) $H(M'_1 \oplus 1) \oplus \ldots \oplus H(M'_n \oplus n)$

○ (C) $M'_i \oplus E_{K_2}(i)$

○ (D) M'

Q5.3 (2 points) Select the correct option for the blank on Step 4:

○ (A) K_2

○ (B) $H(M'_1 \oplus 1) \oplus \ldots \oplus H(M'_n \oplus n)$

○ (C) $M'_i \oplus E_{K_2}(i)$

○ (D) M'

Q5.4 (5 points) Explain how modifying an arbitrary ciphertext block prevents recovery of ***any block*** of the original message.

HINT: Show that we cannot recover K_2 if any ciphertext block is modified.

Q5.5 (5 points) EvanBot wonders if it's really necessary to have the hash function used in Step 3, and decides to replace Step 3 with this new step:

3. EvanBot adds the block $(M'_1 \oplus 1) \oplus (M'_2 \oplus 2) \oplus \ldots \oplus (M'_n \oplus n) \oplus K_2$ to the end of M'.

Show that it is possible to tamper with the order of the message blocks, i.e. by swapping two blocks. Note that "tamper" means the message will be decrypted to something different, but not all blocks will turn to garbage (i.e. not "all or nothing").

Q5.6 (4 points) Does the original all-or-nothing scheme (from the beginning of the question) provide integrity?

○ (A) Yes ○ (B) No

Explain why or why not.

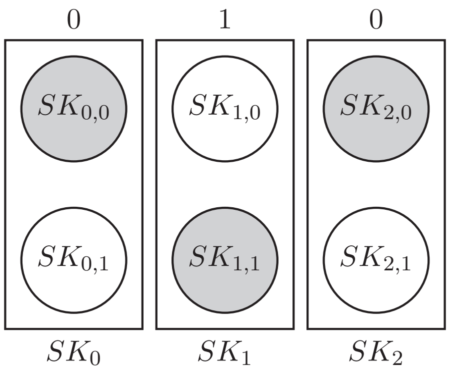
Q6 *Cryptography: One-Time Signatures* (24 points)

One-time signatures are a class of digital signatures that can only sign a single message before becoming insecure.

Consider a scheme to sign a n-bit message m using a hash function H with 256 bit output:

1. Generate n pairs of random 256-bit numbers as the private key, denoted SK_i for the i-th pair. $SK_{i,j}$ for $j = 0$ or $j = 1$ represents the first or second item in the pair, respectively.
2. Publish the public key as the list of all hashed secret key pairs: $PK_{i,j} = H(SK_{i,j})$.
3. To sign a n-bit message m, we consider its binary representation: $m_0, m_1, \dots, m_{n-1}$. The signature $S = (S_0, S_1, \dots, S_{n-1})$ is defined as $S_i = SK_{i,m_i}$.

In other words, for each bit m_i, if $m_i = 0$ we choose the first item from the i-th secret key pair. Otherwise, we choose the second item.



Pictured: Signing the binary message "010". The final signature is $(SK_{0,0}, SK_{1,1}, SK_{2,0})$.

Q6.1 (4 points) Alice has received a message M from Bob, with the one-time signature S using key PK. Select the correct option for verifying whether this signature is valid.

○ (A) Verify that $H(S_i) = PK_{i,m_i}$ for $0 \leq i < n$

○ (B) Verify that $S_i = H(PK_{i,m_i})$ for $0 \leq i < n$

○ (C) Verify that $S_i = PK_{i,m_i}$ for $0 \leq i < n$

○ (D) Invert H to verify $H^{-1}(PK_{i,m_i}) = S_{i,m_i}$

Alice and Bob forget that these signatures are one-time use only and accidentally sign two different n-bit messages with the same secret key: a message of all zeroes, and a message of all ones.

Q6.2 (8 points) Explain how Eve can forge **arbitrary** n-bit messages using (possibly not all of) Alice's public key and the two previous signatures.

Consider a different situation: Alice sends two different signed messages, but they only differ in b bits instead of differing in all n bits.

Q6.3 (6 points) How many new messages (excluding those already sent by Alice) can Eve forge in this new situation?

○ (A) 0 ○ (B) $2^b - 2$ ○ (C) $2^{n-b} - 2$ ○ (D) $2^n - 2$

Explain your reasoning.

Q6.4 (6 points) Alice and Bob want to reduce the size of the public keys for an individual one-time signature scheme, without reducing its security.

Design a scheme that reduces the size of a one-time signature public key to exactly 256 bits.

HINT: The constraint only applies to the publicly trusted public key. The signature itself may or may not have additional information included that does not have to be constant space.

HINT: The output of H is 256 bits.

Describe how to construct the new public key:

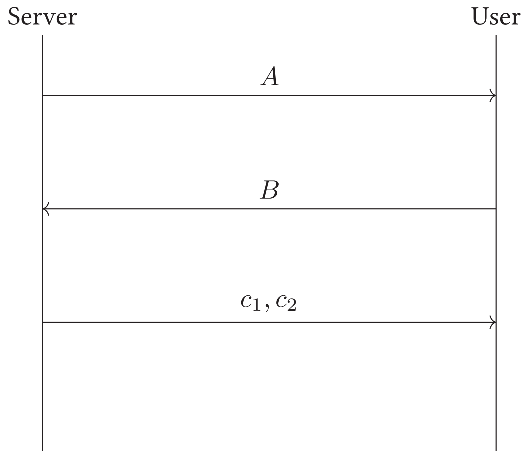
Describe how to produce a signature using this new scheme:

Q7 *Cryptography: Oblivious Transfer* **(26 points)**

A user of a popular document storage website wishes to access one of two possible documents (labeled m_1 and m_2), without the server knowing which document was accessed. They decide to use an **oblivious transfer** scheme. We will consider a 1-of-2 oblivious transfer scheme, in which the user accesses either m_1 or m_2 (but not both).

Clarification during exam: You may assume Enc refers to an IND-CPA symmetric encryption scheme.

Clarification during exam: i refers to the document the user wishes to access (if $i = 1$, they access m_1, if $i = 2$, they access m_2).



1. The server chooses a random number $a \mod p$ and sends $A = g^a \mod p$.

2. The user chooses a random number b. If $i = 1$, they send $B = g^b \mod p$. If $i = 2$, they send $B = Ag^b \mod p$.

3. The server computes $k_1 = B^a$ and $k_2 = \left(\frac{B}{A}\right)^a$ Next, they send $c_1 = \mathsf{Enc}(k_1, m_1)$ and $c_2 = \mathsf{Enc}(k_2, m_2)$.

4. [Answer to Q7.5]

Q7.1 (2 points) When $i = 1$, what value does the server derive for k_1?

○ (A) $g^{ab} \mod p$, ○ (B) $g^{ab-a^2} \mod p$ ○ (C) $g^{ab+a^2} \mod p$ ○ (D) $Ag^{ab} \mod p$

Q7.2 (2 points) When $i = 1$, what value does the server derive for k_2?

○ (A) $g^{ab} \mod p$, ○ (B) $g^{ab-a^2} \mod p$ ○ (C) $g^{ab+a^2} \mod p$ ○ (D) $Ag^{ab} \mod p$

Q7.3 (2 points) When $i = 2$, what value does the server derive for k_1?

○ (A) $g^{ab} \mod p$, ○ (B) $g^{ab-a^2} \mod p$ ○ (C) $g^{ab+a^2} \mod p$ ○ (D) $Ag^{ab} \mod p$

Q7.4 (2 points) When $i = 2$, what value does the server derive for k_2?

○ (A) $g^{ab} \mod p$, ○ (B) $g^{ab-a^2} \mod p$ ○ (C) $g^{ab+a^2} \mod p$ ○ (D) $Ag^{ab} \mod p$

Q7.5 (4 points) Give an expression for k_i (the i-th document key) in terms of (possibly not all of) A, B, b, c_i, g, p.

Q7.6 (3 points) Which option best describes why the server is not able to tell which document the user has chosen to read?

◯ (A) Since b is randomly chosen, $g^b \mod p$ will look random to the server.

◯ (B) The server cannot solve the discrete logarithm to recover a from $g^a \mod p$.

◯ (C) The server cannot tell the difference between $g^b \mod p$ and $Ag^b \equiv g^{a+b} \mod p$.

Q7.7 (3 points) Which option best describes why the user is not able to read more than a single document per request?

◯ (A) The encryption function used is IND-CPA secure.

◯ (B) The user cannot derive $g^{ab \pm a^2} \mod p$ despite knowing $g^{ab} \mod p$.

◯ (C) The server will only derive k_1 or k_2, but not both.

The user wants to expand their document storage from 2 to 3 documents, and need to update their oblivious transfer scheme to account for this. They know that Steps 1 and 4 from the previous scheme will stay the same, but still need to update Steps 2 and 3.

Q7.8 (4 points) The new Step 2 is as follows:

"The user chooses a random number b. If $i = 1$, they send $B = g^b \mod p$. If $i = 2$, they send $B = Ag^b \mod p$. If $i = 3$, they send $B =$ ________________."

Give an expression for the blank in this new step.

Q7.9 (4 points) The new Step 3 is as follows:

"The server computes $k_1 = B^a$, $k_2 = \left(\frac{B}{A}\right)^a$, and $k_3 =$ ________________. Next, they send $c_1 = \mathsf{Enc}(k_1, m_1)$, $c_2 = \mathsf{Enc}(k_2, m_2)$, and $c_3 = \mathsf{Enc}(k_3, m_3)$."

Give an expression for the blank in this new step.

(this page is intentionally blank)

Nothing on this page will affect your grade in any way.

(Optional) Post-exam Activity: Battleship

EvanBot wants to play battleship with CS 161 students. EvanBot has chosen the positions of the ships on their board, but won't reveal them until after the exam. However, they have published a SHA−3 hash of their board and ship locations so you can verify they haven't changed them later:

H(board) = `0xd726c59246ede23df594586246d5983924d00a06a7736ab67aa89c1db1461688`

Now you have the chance to try and hit their ships. On the grid below, mark **five** squares with an `X` where you believe EvanBot has placed their ships. After the exam, EvanBot will reveal their board and you can see how many ships you hit.

	1	2	3	4	5	6	7	8	9	10
A										
B										
C										
D										
E										
F										
G										
H										
I										
J										