Nicholas & Peyrin
Summer 2021

CS 161 Computer Security

Final

For questions with **circular bubbles**, you may select exactly *one* choice on Examtool.

○ Unselected option

● Only one selected option

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

■ You can select

■ multiple squares

For questions with a **large box**, you need to write your answer in the text box on Examtool.

There is an appendix at the end of this exam, containing descriptions of all C functions used on this exam.

You have 170 minutes, plus a 10-minute buffer for distractions or technical difficulties, for a total of 180 minutes. There are 12 questions of varying credit (200 points total).

The exam is open note. You can use an unlimited number of handwritten cheat sheets, but you must work alone.

Clarifications will be posted on Examtool.

Q1 *MANDATORY – Honor Code* (5 points)

Read the following honor code and type your name on Examtool.

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 and a corresponding notch on Nick's Stanley Fubar demolition tool.

Q2 *True/false* (30 points)

Each true/false is worth 2 points.

Q2.1 True or False: TLS with Diffie-Hellman is still vulnerable to man-in-the-middle attackers by intercepting and performing two separate key exchanges with both sides.

○ True ○ False

Q2.2 True or False: In TLS, an attacker cannot change the client and server random values R_b and R_s sent at the beginning of the exchange undetected, even though the client and server have not yet agreed on a set of symmetric keys.

○ True ○ False

Q2.3 True or False: Referrer validation is a valid defense against reflected XSS attacks.

Clarification during exam: This question has been dropped. All students will receive credit on this question.

○ True ○ False

Q2.4 True or False: Hybrid encryption typically uses symmetric encryption to encrypt the plaintext and asymmetric encryption to encrypt a symmetric key.

○ True ○ False

Q2.5 True or False: If the first and last node in a Tor circuit learn that they are part of the same circuit and collude with one another, they are able to deanonymize the Tor user, even if the middle node is honest.

○ True ○ False

Q2.6 True or False: One advantage of pointer authentication over stack canaries is that pointer authentication is harder to brute-force than stack canaries.

○ True ○ False

Q2.7 True or False: While developing a website, the programmer leaves the server password hidden in the website's HTML. This violates Shannon's Maxim.

○ True ○ False

Q2.8 True or False: Most CAPTCHAs are set up to distinguish good bots, such as web crawlers, from malicious bots.

○ True ○ False

Q2.9 True or False: Cursorjacking is a UI attack that relies on creating a fake cursor that is more prominent or visible than the real cursor.

○ True ○ False

Q2.10 True or False: In a program that has stack canaries enabled, the stack canary takes on the same value across multiple executions of the program.

◯ True ◯ False

Q2.11 True or False: If Alice encrypts a message with AES-CBC, but instead of using completely random IVs, she uses R, $R+1$, $R+2$, and so on, where R is a random value that she chose once, this scheme is IND-CPA secure.

◯ True ◯ False

Q2.12 True or False: DNSSEC uses TLS to protect against attacks when transmitting packets.

◯ True ◯ False

Q2.13 True or False: SHA-256 is a good hash function to use when hashing passwords.

◯ True ◯ False

Q2.14 True or False: A firewall does not defend against a trusted party inside the firewall that becomes malicious and attempts to breach other computers within the network.

◯ True ◯ False

Q2.15 True or False: One weakness of one-time authentication codes, such as those sent to a user's cell phone number, is that they are still vulnerable to a transient phishing attack.

◯ True ◯ False

Q2.16 (0 points) True or False: `EvanBot is a real bot.`

◯ True ◯ False

Q3 *Piazza Policy* (18 points)

Q3.1 (5 points) Which of the following URLs have the same origin as `http://piazza.com/`? Select all that apply.

☐ (A) `https://piazza.com/`

☐ (B) `http://piazza.com:614/`

☐ (C) `http://web.piazza.com/`

☐ (D) `http://piazza.com/run`

☐ (E) `http://aplaza.com/`

☐ (F) None of the above

Q3.2 (3 points) If the script `<script src="http://cs161.org/tracking.js"></script>` is included in `http://piazza.com/`, which of these pages can the script modify?

○ (G) `http://piazza.com/`

○ (H) `http://bank.com/`

○ (I) `http://cs161.org/`

○ (J) All of the above

○ (K) None of the above

○ (L) —

Q3.3 (5 points) In which of the following contexts would the contents of a cookie with the HttpOnly flag be sent? Select all that apply.

☐ (A) A user clicks on a link that sends a request over HTTP to the domain of the cookie

☐ (B) A user clicks on a link that sends a request over HTTPS to the domain of the cookie

☐ (C) JavaScript is used to send a request over HTTP to the domain of the cookie

☐ (D) JavaScript is used to read the value of the cookie and send its contents to a different domain

☐ (E) A page includes an `<img>` tag loading an image using HTTP from the domain of the cookie

☐ (F) None of the above

Q3.4 (5 points) `http://evanbot.piazza.com` is setting a cookie. Which of these cookie attributes can `http://evanbot.piazza.com` set (without being rejected by the browser) so that the cookie gets sent on a request to `http://web.piazza.com/pictures`? Select all that apply.

☐ (G) `domain=evanbot.piazza.com; path=/pictures`

☐ (H) `domain=piazza.com; path=/pictures`

☐ (I) `domain=com; path=/pictures`

☐ (J) `domain=evanbot.piazza.com`

☐ (K) `domain=piazza.evanbot.com; path=/pictures`

☐ (L) None of the above

Q4 *Coffee-Shop Attacks* (17 points)

Dr. Yang comes to MoonBucks and tries to connect to the network in the coffee shop. Dr. Yang and `http://www.piazza.com` are communicating through TCP. Mallory is an on-path attacker.

Q4.1 (5 points) Which of the following protocols are used when Dr. Yang first connects to the Wi-Fi network and visits `http://www.piazza.com`? Assume any caches are empty. Select all that apply.

☐ (A) CSRF ☐ (C) DNS (or DNSSEC) ☐ (E) DHCP

☐ (B) IP ☐ (D) HTTP ☐ (F) None of the above

Q4.2 (3 points) Suppose Mallory spoofs a packet with a valid, upcoming sequence number to inject the malicious message into the connection. Would this affect other messages in the connection?

○ (G) Yes, because the malicious message replaces some legitimate message

○ (H) Yes, because future messages will arrive out of order

○ (I) No, because on-path attackers cannot inject packets into a TCP connection

○ (J) No, because TCP connections are encrypted

○ (K) ——

○ (L) ——

Q4.3 (3 points) To establish a TCP connection, Dr. Yang first sends a SYN packet with Seq $= 980$ to the server and receives a SYN-ACK packet with Seq $= 603$; Ack $= 981$. What packet should Dr. Yang include in the next packet to complete the TCP handshake?

○ (A) SYN-ACK packet with Seq $= 981$; Ack $= 604$

○ (B) SYN-ACK packet with Seq $= 604$; Ack $= 981$

○ (C) ACK packet with Seq $= 981$; Ack $= 604$

○ (D) ACK packet with Seq $= 604$; Ack $= 981$

○ (E) Nothing to send, because the TCP handshake is already finished.

○ (F) ——

Q4.4 (3 points) Immediately after the TCP handshake, Mallory injects a valid RST packet to the server. Next, Mallory spoofs a SYN packet from Dr. Yang to the server with headers Seq $= X$. The server responds with a SYN-ACK packet with Seq $= Y$; Ack $= X + 1$. What is the destination of this packet?

○ (G) Dr. Yang ○ (H) The server

◯ (I) Mallory

◯ (K) ——

◯ (J) None of the above

◯ (L) ——

Q4.5 (3 points) Which of the following network attackers would be able to perform the same attacks as Mallory?

Clarification during exam: By "perform the same attacks," we mean "reliably perform the same attacks."

◯ (A) A MITM attacker between Dr. Yang and the server

◯ (B) An off-path attacker

◯ (C) All of the above

◯ (D) None of the above

◯ (E) ——

◯ (F) ——

Q5 *Dual Asymmetry* (15 points)

Alice wants to send two messages M_1 and M_2 to Bob, but they do not share a symmetric key.

Clarification during exam: Assume that p is a large prime and that g is a generator $\bmod p$, like in ElGamal. Assume that all computations are done modulo p in Scheme A.

Q5.1 (3 points) Scheme A: Bob publishes his public key $B = g^b$. Alice randomly selects r from 0 to p-2. Alice then sends the ciphertext $(R, S_1, S_2) = (g^r, M_1 \times B^r, M_2 \times B^{r+1})$.

Select the correct decryption scheme for M_1:

○ (A) $R^{-b} \times S_1$

○ (B) $R^b \times S_1$

○ (C) $B^{-b} \times S_1$

○ (D) $B^b \times S_1$

○ (E) —

○ (F) —

Q5.2 (3 points) Select the correct decryption scheme for M_2:

○ (G) $B^{-1} \times R^{-b} \times S_2$

○ (H) $B \times R^{-b} \times S_2$

○ (I) $B^{-1} \times R^b \times S_2$

○ (J) $B^{-1} \times R \times S_2$

○ (K) —

○ (L) —

Q5.3 (4 points) Is Scheme A IND-CPA secure? If it is secure, briefly explain why (1 sentence). If it is not secure, briefly describe how you can learn something about the messages.

Clarification during exam: For Scheme A, in the IND-CPA game, assume that a single plaintext is composed of two parts, M_1 and M_2.

○ (A) Secure

○ (B) Not secure

○ (C) —

○ (D) —

○ (E) —

○ (F) —

Q5.4 (5 points) Scheme B: Alice randomly chooses two 128-bit keys K_1 and K_2. Alice encrypts K_1 and K_2 with Bob's public key using RSA (with OAEP padding) then encrypts both messages with AES-CTR using K_1 and K_2. The ciphertext is $\text{RSA}(\mathsf{PK}_{\mathsf{Bob}}, K_1 \| K_2), \text{Enc}(K_1, M_1), \text{Enc}(K_2, M_2)$.

Which of the following is required for Scheme B to be IND-CPA secure? Select all that apply.

☐ (G) K_1 and K_2 must be different

☐ (H) A different IV is used each time in AES-CTR

☐ (I) M_1 and M_2 must be different messages

☐ (J) M_1 and M_2 must be a multiple of the AES block size

☐ (K) M_1 and M_2 must be less than 128 bits long

☐ (L) None of the above

Q6 *Under Pressure* (16 points)

Alice and Bob are communicating large pieces of secret data with each other using an IND-CPA secure encryption scheme, but they are being slowed down by their connection! They decide to use compression to improve the amount of data they can send.

Consider the following properties of compression algorithms:

- Compression algorithms reduce the length of the input.
- High-entropy (random-looking) data may only have its length reduced slightly, or not at all
- Low-entropy (predictable) data can often have its length reduced considerably

Q6.1 (3 points) Alice and Bob consider first encrypting and then compressing their data. They send $\mathsf{compress}(\mathsf{Enc}(K, M))$, where the input to the compression algorithm is the output of the encryption algorithm. Provide **one** reason why this might be a bad idea.

Q6.2 (5 points) Realizing their mistake, Alice and Bob instead decide to compress their data before encrypting it. They send $\mathsf{Enc}(K, \mathsf{compress}(M))$, where the input to the encryption algorithm is the output of the compression algorithm.

TRUE or FALSE: This combination of compress-then-encrypt is IND-CPA secure. If you answer True, briefly justify your answer (no formal proof needed). If you answer False, describe how an adversary could win the IND-CPA game with probability greater than 0.5.

○ (G) True ○ (H) False ○ (I) —— ○ (J) —— ○ (K) —— ○ (L) ——

For the rest of this question, consider a compress-then-encrypt algorithm with the following properties:

- Assume that Alice and Bob's messages consist of byte strings.
- When compressing, any chain of m consecutive, identical runs of n bytes (total length mn) are compressed into a single run of length n. Runs consisting of a single byte ($n = 1$) are included.
- Encryption preserves the exact length of the message (no padding is used).

For example, the following sequence of bytes:

Index	0	1	2	3	4	5	6	7	8	9
Byte	`0x00`	`0x11`	`0x22`	`0x33`	`0x44`	`0x44`	`0x55`	`0x66`	`0x77`	`0x77`

would be compressed to a message of length 8, since there are 2 1-byte runs in positions 4–5 and 8–9.

Similarly, the following sequence of bytes:

Index	0	1	2	3	4	5	6	7	8	9
Byte	`0xaa`	`0xbb`	`0xaa`	`0xbb`	`0xcc`	`0xdd`	`0xcc`	`0xdd`	`0xee`	`0xff`

would be compressed to 6 bytes, since there are two 2-byte runs in positions 0–3 and two 2-byte runs in positions 4–7.

Q6.3 (5 points) Alice sends the same message $M =$ secret repeatedly to Bob, using the compression algorithm from above for a compress-then-encrypt scheme.

Assume that Mallory doesn't have complete control over the messages that Alice sends, but Mallory is able to *append* to the sent messages before they are compressed and encrypted.

For example, if Alice repeatedly sends the message $M =$ secret, Mallory can force the compressed and encrypted message to be $M' =$ secret$\|x$, where x is chosen by Mallory. She can do this repeatedly for any value of x that she chooses.

If Mallory can observe all resulting ciphertexts from the compress-then-encrypt algorithm as they are transmitted to Bob, devise a way to recover the **last** byte of secret.

Clarification during exam: Assume that Alice's messages will never produce ambiguous behavior when being compressed.

Clarification during exam: Assume that the encryption algorithm preserves the exact length of the plaintext.

Q6.4 (3 points) Assume that Mallory has a method for recovering the last byte of the message. If secret is 32 bytes long and there are 256 possible values for a byte, what is the most number of messages Alice has to send (that Mallory can append to) in order to learn the entire secret?

- ◯ (G) 2^{128}
- ◯ (I) 32^{256}
- ◯ (K) 256^{32}
- ◯ (H) 2^{256}
- ◯ (J) 256×32
- ◯ (L) None of the above

Q7 *TLS Secret Chaining* **(12 points)**

Recall that TLS with RSA does not provide forward secrecy. An attacker who has recorded past connections and then stolen the server's private key can decrypt any past connection.

Consider a modified TLS scheme. RSA is used for the first connection. In future connections, the premaster secret is instead encrypted using the cipher key from the previous connection. Assume this encryption is IND-CPA secure.

The server and client perform n TLS connections. The first connection is numbered C_1, and the most recent connection is numbered C_n. The attacker records some of these connections and then steals the server's private key. For each set of recorded connections, select all connections the attacker can decrypt.

Clarification during exam: Assume that ranges of connections are inclusive. For example, "All connections between C_1 and C_n" would include C_1 and C_n.

Q7.1 (3 points) The attacker records all connections except C_1.

◯ (A) Connection C_1 only

◯ (B) Connections C_1 and C_2 only

◯ (C) All connections except C_1

◯ (D) All connections between C_1 and C_{n-1}

◯ (E) All connections between C_1 and C_n

◯ (F) None of the above

Q7.2 (3 points) The attacker records all connections except C_n.

◯ (G) Connection C_1 only

◯ (H) Connections C_1 and C_2 only

◯ (I) All connections except C_1

◯ (J) All connections between C_1 and C_{n-1}

◯ (K) All connections between C_1 and C_n

◯ (L) None of the above

Q7.3 (3 points) The attacker records all connections except C_2.

◯ (A) Connection C_1 only

◯ (B) Connections C_1 and C_2 only

◯ (C) All connections except C_1

◯ (D) All connections between C_1 and C_{n-1}

◯ (E) All connections between C_1 and C_n

◯ (F) None of the above

Q7.4 (3 points) Suppose we modify the TLS scheme from above. In every connection, the client encrypts their random number R_b with RSA before sending it to the server. (Recall that in regular TLS, R_b is sent with no encryption.)

Does this scheme provide forward secrecy?

◯ (G) Yes, because R_b is needed to generate the symmetric keys

◯ (H) Yes, because an attacker can perform a replay attack

○ (I) No, because the attacker can still learn R_b after stealing the private key

○ (J) No, because only Diffie-Hellman TLS provides forward secrecy

○ (K) ——

○ (L) ——

Q8 *Intrusion Detection Scenarios* (12 points)

For each scenario below, select the best detector or detection method for the attack.

Q8.1 (3 points) The attacker constructs a path traversal attack with URL escaping: `%2e%2e%2f%2e%2e%2f`.

- ◯ (A) NIDS, because of interpretation issues
- ◯ (B) NIDS, because of cost
- ◯ (C) HIDS, because of interpretation issues
- ◯ (D) HIDS, because of cost
- ◯ (E) —
- ◯ (F) —

Q8.2 (3 points) The attacker is attacking a large network with hundreds of computers, and a detector must be installed as quickly as possible.

- ◯ (G) NIDS, because of interpretation issues
- ◯ (H) NIDS, because of cost
- ◯ (I) HIDS, because of interpretation issues
- ◯ (J) HIDS, because of cost
- ◯ (K) —
- ◯ (L) —

Q8.3 (3 points) The attacker constructs an attack that is encrypted with HTTPS.

- ◯ (A) NIDS, because of interpretation issues
- ◯ (B) NIDS, because of cost
- ◯ (C) HIDS, because of interpretation issues
- ◯ (D) HIDS, because of cost
- ◯ (E) —
- ◯ (F) —

Q8.4 (3 points) The attacker constructs a buffer overflow attack using shellcode they found online in a database of common attacks.

- ◯ (G) Signature-based
- ◯ (H) Specification-based
- ◯ (I) Anomaly-based
- ◯ (J) Behavioral
- ◯ (K) —
- ◯ (L) —

Q9 *To The Moon* (15 points)

ToTheMoon Bank has just created an online banking system. When a user wants to complete a transfer, they follow these steps:

1. The user logs in by making a POST request with their username and password.
2. The server sets a cookie with `name=auth_user` and `value=$token`, where `$token` is a session token specific to the user's login session.
3. The user initiates a transfer by making a GET request to `https://tothemoonbank.com/transfer?amount=$amount&to=$user`, replacing `$amount` and `$user` with the intended amount and recipient. Transfers use a parameterized SQL query.
4. The server runs the SQL query `SELECT username FROM users WHERE session_token = '$token'`, replacing `token` with the value of the cookie. The server does not use parameterized SQL or any input sanitization.

Q9.1 (4 points) Which of the following attacks are possible in this system? Select all that apply.

☐ (A) SQL injection

☐ (B) ROP attack

☐ (C) CSRF attack

☐ (D) Path traversal attack

☐ (E) None of the above

☐ (F) ——

Q9.2 (4 points) Mallory is a malicious user with an account on ToTheMoon Bank. Mallory creates a malicious link `https://tothemoonbank.com/transfer?amount=100&to=Mallory`.

Which of the following scenarios would cause Alice to send $100 to Mallory? Select all that apply.

☐ (G) Alice clicks on the malicious link when Alice is not logged into the bank

☐ (H) Alice clicks on the malicious link when Alice is logged into the bank

☐ (I) Alice visits Mallory's website, which has an `img` tag loading the malicious link, when Alice is not logged into the bank

☐ (J) Alice visits Mallory's website, which has an `img` tag loading the malicious link, when Alice is logged into the bank

☐ (K) None of the above

☐ (L) ——

Q9.3 (4 points) Suppose Step 4 is modified. To initiate a transfer, instead of making a GET request, the user makes a POST request to `https://tothemoonbank.com/transfer` with the amount and recipient in the POST body.

Which of the following scenarios would cause Alice to send $100 to Mallory? Select all that apply.

Clarification during exam: The malicious link in the answer choices should be `https://tothemoonbank.com/transfer?amount=100&to=Mallory`.

☐ (A) Alice clicks on the malicious link when Alice is not logged into the bank

☐ (B) Alice clicks on the malicious link when Alice is logged into the bank

☐ (C) Alice visits Mallory's website, which has an `img` tag loading the malicious link, when Alice is not logged into the bank

☐ (D) Alice visits Mallory's website, which has an `img` tag loading the malicious link, when Alice is logged into the bank

☐ (E) None of the above

☐ (F) ——

Q9.4 (3 points) Which user inputs might be vulnerable to SQL injection? Select all that apply.

☐ (G) `amount` parameter

☐ (H) `to` parameter

☐ (I) Value of the `auth_user` cookie

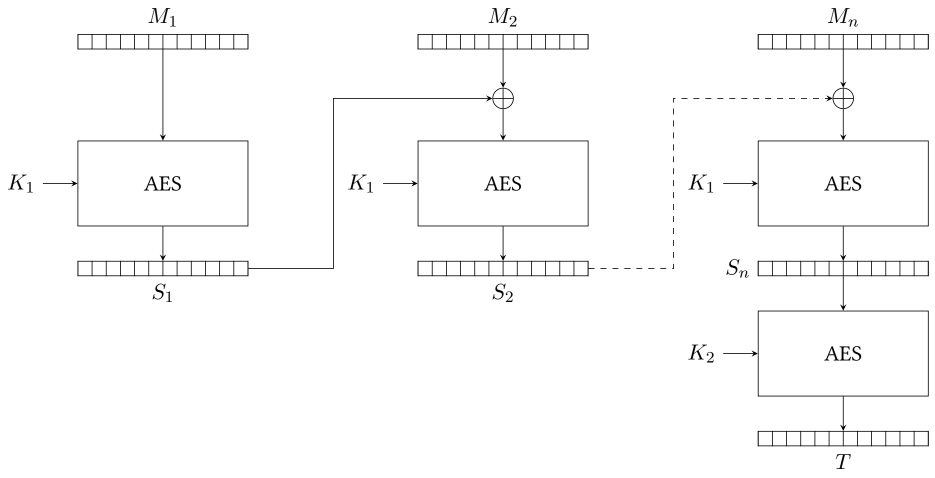
☐ (J) None of the above

☐ (K) ——

☐ (L) ——

Q10 *AES-EMAC* (19 points)

Consider AES-EMAC, which is another scheme for generating secure MACs.



Q10.1 (2 points) TRUE or FALSE: Given only T, an attacker can generate a valid MAC for $M||M'$, for an M' of the attacker's choosing.

◯ TRUE ◯ FALSE

Q10.2 (2 points) TRUE or FALSE: Given only T and K_1, an attacker can generate a valid MAC for $M||M'$, for an M' of the attacker's choosing.

◯ TRUE ◯ FALSE

Q10.3 (2 points) TRUE or FALSE: Given only T, K_1, and K_2, an attacker can generate a valid MAC for $M||M'$, for an M' of the attacker's choosing.

◯ TRUE ◯ FALSE

Q10.4 (2 points) TRUE or FALSE: Given the output T and the secret keys K_1 and K_2, the entire original message M can be reconstructed.

◯ TRUE ◯ FALSE

For the rest of the question, regardless of your answer to the previous parts, assume that the output is $S_1||S_2\ldots||S_n||T$.

Q10.5 (4 points) Which values are needed to recover the entire original message? Select all that apply.

☐ (A) S_i for all $1 \leq i \leq n$

☐ (D) K_2

☐ (B) T

☐ (E) None of the above

☐ (C) K_1

☐ (F) ——

Q10.6 (3 points) Which of these equations is correct for calculating a plaintext block M_i given the output and both keys K_1 and K_2?

○ (G) $M_i = \text{Dec}(K_1, S_i) \oplus S_{i-1}$

○ (H) $M_i = \text{Dec}(K_1, S_i \oplus S_{i-1})$

○ (I) $M_i = \text{Dec}(K_2, S_i) \oplus S_{i-1}$

○ (J) $M_i = \text{Dec}(K_2, S_i \oplus S_{i-1})$

○ (K) $M_i = \text{Enc}(K_1, S_i) \oplus S_{i-1}$

○ (L) $M_i = \text{Enc}(K_1, S_i \oplus S_{i-1})$

Q10.7 (4 points) Select all true statements about this scheme.

☐ (A) To compute AES-EMAC on a message, the message must first be padded to a multiple of the block size

☐ (B) Encryption can be parallelized

☐ (C) Decryption can be parallelized

☐ (D) AES-EMAC is IND-CPA secure

☐ (E) None of the above

☐ (F) ——

Q11 *DNS Lookups* (15 points)

EvanBot performs a lookup for the IP address of `toon.cs161.org`. For each DNS or DNSSEC record, determine which name server sent the record.

Q11.1 (3 points) `A` type record with the IP address of `toon.cs161.org`

- ○ (A) root name server
- ○ (B) `.org` name server
- ○ (C) `cs161.org` name server
- ○ (D) None of the above
- ○ (E) —
- ○ (F) —

Q11.2 (3 points) `A` type record with the IP address of the `cs161.org` name server

- ○ (G) root name server
- ○ (H) `.org` name server
- ○ (I) `cs161.org` name server
- ○ (J) None of the above
- ○ (K) —
- ○ (L) —

Q11.3 (3 points) `A` type record with the IP address of `cs161.org` (not the name server)

- ○ (A) root name server
- ○ (B) `.org` name server
- ○ (C) `cs161.org` name server
- ○ (D) None of the above
- ○ (E) —
- ○ (F) —

Q11.4 (3 points) `DNSKEY` type record with the public key of the `cs161.org` name server

- ○ (G) root name server
- ○ (H) `.org` name server
- ○ (I) `cs161.org` name server
- ○ (J) None of the above
- ○ (K) —
- ○ (L) —

Q11.5 (3 points) `DS` type record with the hash of the `.org` name server's public key

- ○ (A) root name server
- ○ (B) `.org` name server
- ○ (C) `cs161.org` name server
- ○ (D) None of the above
- ○ (E) —
- ○ (F) —

Q12 *Wallet Management* (26 points)

Consider the following vulnerable C code:

```c
#include <stdio.h>
#include <stdlib.h>

struct wallet {
    char owner[4];   /* 4 bytes. */
    int amt;         /* 4 bytes. */
};

int main(void) {
    int wallet_idx = 0;
    struct wallet wals[8];
    char buf[16];

    while (1) {
        /* Get wallet index. */
        printf("Enter wallet index:\n");
        fgets(buf, 16, stdin);
        int wallet_idx = atoi(buf);
        if (wallet_idx < 0) {
            /* Exit loop if invalid index. */
            break;
        }

        /* Update dollar amount. */
        printf("Enter dollar amount:\n");
        fgets(buf, 16, stdin);
        wals[wallet_idx].amt = atoi(buf);

        /* Read owner. */
        printf("Enter owner name:\n");
        gets(wals[wallet_idx].owner);
    }

    return 0;
}
```

Assume you are on a little-endian 32-bit x86 system. Assume that there is no compiler padding or additional saved registers in all subparts.

Clarification during exam: The `atoi` function converts a string to an integer. For example, `atoi("3")` would return the integer value 3.

Q12.1 (4 points) For the first subpart, assume that **no memory safety defenses** are enabled.

Let `SHELLCODE` be a 24-byte malicious shellcode. If the address of `wals` is 0x9fffcad0, provide an input as a series of Python `print` statements that would cause your program to execute malicious shellcode.

For example, a sequence of non-malicious inputs that updates wallet 0 without exiting the loop would be:

```
print('0')
print('1000')
print('NN')
```

Write your answer in Python 2 syntax (just like in Project 1).

For the remaining parts of this question, assume that **stack canaries are enabled**.

Q12.2 (3 points) Complete the stack diagram with stack canaries enabled. Each row represents 4 bytes.

Parts (2a), (2b), and (2c):

RIP of `main`
(2a)
(2b)
(2c)
(3a)
(3b)
(3c)
(3d)

◯ (G) (2a) - SFP of `main`; (2b) - `wallet_idx`; (2c) - canary

◯ (H) (2a) - SFP of `main`; (2b) - canary; (2c) - `wallet_idx`

◯ (I) (2a) - canary; (2b) - SFP of `main`; (2c) - `wallet_idx`

◯ (J) (2a) - canary; (2b) - `wallet_idx`; (2c) - SFP of `main`

◯ (K) (2a) - `wallet_idx`; (2b) - SFP of `main`; (2c) - canary

◯ (L) (2a) - `wallet_idx`; (2b) - canary; (2c) - SFP of `main`

Q12.3 (3 points) Parts (3a), (3b), (3c), and (3d):

◯ (A) (3a) - `wals[0].amt`; (3b) - `wals[0].owner`; (3c) - `wals[1].amt`; (3d) - `wals[1].owner`

◯ (B) (3a) - `wals[0].owner`; (3b) - `wals[0].amt`; (3c) - `wals[1].owner`; (3d) - `wals[1].amt`

◯ (C) (3a) - `wals[7].amt`; (3b) - `wals[7].owner`; (3c) - `wals[6].amt`; (3d) - `wals[6].owner`

◯ (D) (3a) - `wals[7].owner`; (3b) - `wals[7].amt`; (3c) - `wals[6].owner`; (3d) - `wals[6].amt`

◯ (E) ——

◯ (F) ——

Q12.4 (3 points) Describe, briefly, a vulnerability on line 19. Additionally, describe a one-line fix to line 19 that would make this code memory-safe.

Q12.5 (5 points) Let `SHELLCODE` be a 24-byte malicious shellcode. If the address of the RIP of `main` is 0xbfeffc80, provide an input as a series of Python `print` statements that would cause your program to execute malicious shellcode.

Q12.6 (4 points) Assume that the call to `gets` on line 31 is replaced with `fread(wals[wallet_idx].owner, 1, 4, stdin)`. Recall that `fread` does stop reading at a newline and does not add a NULL terminator.

EvanBot thinks that this code is no longer exploitable because the `gets` function is no longer used. **Assume that your 24-byte shellcode must be written in a contiguous block of memory.** Is EvanBot correct? If you answer Yes, describe why you can no longer exploit this code. If you answer No, describe how you would write your shellcode to a contiguous block of memory.

◯ (G) Yes ◯ (H) No ◯ (I) —— ◯ (J) —— ◯ (K) —— ◯ (L) ——

Q12.7 (4 points) This part is independent of the previous subpart, but assume that stack canaries are still enabled. Which of the following actions would individually prevent the attacker from executing malicious shellcode (not necessarily using the exploit from above)?

Clarification during exam: The answer choice "Enabling pointer authentication" has been dropped. All students will receive credit for that answer choice only.

☐ (A) Enabling non-executable pages in addition to stack canaries

☐ (B) Enabling pointer authentication

☐ (C) Replacing the call to `gets` on line 31 with `fgets(wals[wallet_idx].owner, 4, stdin)`

☐ (D) Swapping the positions of the `owner` and `amt` fields in the `wallet` struct

☐ (E) None of the above

☐ (F) ——

C Function Definitions

```
int printf(const char *format, ...);

       printf() produces output according to the format string format.

char *gets(char *s);

       gets()  reads  a  line from stdin into the buffer pointed to by s until
       either a terminating newline or EOF, which it replaces with a null byte
       ('\0').

char *fgets(char *s, int size, FILE *stream);

       fgets()  reads in at most one less than size characters from stream and
       stores them into the buffer pointed to by s.  Reading  stops  after  an
       EOF  or a newline.  If a newline is read, it is stored into the buffer.
       A terminating null byte ('\0') is stored after the  last  character  in
       the buffer.

size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream);

       The function fread() reads nmemb items of data, each size bytes long,
       from the stream pointed to by stream, storing them at the location
       given by ptr.

       Note that fread() does not add a null byte after input.
```