Weaver Fall 2020 — CS 161 Computer Security — Final Exam

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

- ◯ Unselected option
- ● Only one selected option

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

- ■ You can select
- ■ multiple squares

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

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 11 questions of varying credit (200 points total).

The Gradescope answer sheet assignment has a time limit of 180 minutes. Do not click "Start Assignment" until you're ready to start the exam. The password to decrypt the PDF is at the top of the answer sheet.

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

Clarifications will be posted at https://cs161.org/clarifications.

Q1 *MANDATORY – Honor Code* (5 points)

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

> 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 not on Nick's staff but on his Stanley Fubar demolition tool.

Solution: Don't worry if you forgot to fill in your name. Everyone gets 5 free points for embracing the suck this semester.

We won't take any points off if you entered something for a subpart that doesn't exist, or if you filled in a text box on a multiple-choice question, or vice-versa. To be consistent, we will not consider any unnecessary writing/bubbling on your exam during grading (pretend it's scratch work).

This is the end of Q1. Leave the remaining subparts of Q1 blank on Gradescope, if there are any. Proceed to Q2 on your answer sheet.

Q2 *True/false* **(34 points)**

Each true/false is worth 2 points.

Q2.1 TRUE or FALSE: A cookie with the `Secure` flag set cannot be exploited in an XSS attack.

○ TRUE ● FALSE

Solution: False. The `Secure` flag prevents a cookie from being sent over an unencrypted HTTP connection, which is unrelated to XSS vulnerabilities. Cookies with the `HttpOnly` flag are secure against XSS attacks, however.

Q2.2 TRUE or FALSE: A stack canary is placed above the local variables but below the rip of a given stack frame in order to defend against buffer overflow vulnerabilities.

Clarification during exam: "below the rip" means "somewhere below the rip," not necessarily directly below the rip.

● TRUE ○ FALSE

Solution: True. Stack canaries defend against buffer overflows because stack buffers reside in the local variables, so an overflow would have to overwrite the stack canary before it overwrites the rip.

Q2.3 TRUE or FALSE: Secure cryptographic hash functions provide IND-CPA confidentiality on a message because they are irreversible.

○ TRUE ● FALSE

Solution: False. The simplest way to reason about this is to note that hashes are deterministic, so they cannot be IND-CPA secure.

More formally, IND-CPA confidentiality provides a stricter sense of security in that *nothing* about the message is learned other than its length. Let H be a secure hash function. Define $H'(x) = H(x)||x_0$—that is, H' produces the output of H plus the first bit of x_0. Because of the properties of H, H' is still one-way (since reversing H' would reverse H) and collision resistant (since a collision on H' is a collision on H), but the first bit of x is learned, violating confidentiality.

Q2.4 Let E be an IND-CPA secure encryption scheme, and $E'(x) = E(x)||\mathsf{len(x)}$. In other words, $E'(x)$ is the ciphertext $E(x)$ concatenated with the length of the plaintext x.

TRUE or FALSE: E' is IND-CPA secure.

● TRUE ○ FALSE

Solution: True. The definition of IND-CPA allows IND-CPA secure schemes to leak the length of the plaintext. (Recall that in the IND-CPA game, the pair of messages in the challenge are of equal length.) Therefore, exposing the length directly is still secure.

Keep in mind that for many IND-CPA schemes, such as AES-CBC or AES-CTR, the attacker can estimate the length of the plaintext using the length of the ciphertext.

Q2.5 True or False: Modern, freely-available computer vision programs have become powerful enough to make CAPTCHAs obsolete.

○ True ● False

Solution: False. CAPTCHAs are most vulnerable to outsourcing attacks, where attackers use real human labor, not computer vision programs, to solve them. CAPTCHAs are also still widely used.

Q2.6 True or False: The Great Firewall of China can inject TCP RST packets to censor connections.

● True ○ False

Solution: True. The Great Firewall is an on-path adversary, so it can see TCP packets and spoof RST packets.

Q2.7 True or False: Modern systems enable stack canaries, W^X, ASLR, and pointer authentication to defend against buffer overflow attacks. This is an example of defense-in-depth.

● True ○ False

Solution: True. Even if your exploit defeats one of the defenses, it may not defeat all of them. This is an example of defense-in-depth.

Q2.8 True or False: It is possible to inspect encrypted HTTPS traffic with a HIDS.

● True ○ False

Solution: True. The HIDS is installed on the end host and can see unencrypted application-layer data.

Q2.9 True or False: It is easier for an off-path attacker to inject messages into a TCP connection if the initial sequence numbers were derived from the current time (with second precision) than if the initial sequence numbers were generated randomly.

● True ○ False

Solution: True. An off-path attacker must guess sequence numbers to inject messages in a TCP connection. Non-random sequence numbers make TCP more vulnerable to an off-path attacker, since the attacker has a greater chance of predicting the sequence numbers.

Q2.10 True or False: TLS is vulnerable to RST injection attacks during the handshake, but not after the handshake is completed.

○ True ● False

Solution: False. A TCP RST packet can still be injected during a TLS connection, since TLS is built on top of TCP.

Q2.11 True or False: Input sanitation helps defend against some SQL injection and XSS attacks.

● True ○ False

Solution: True. SQL injection and XSS rely on user input being treated as code, so input sanitation would stop some (but not all) attacks.

Q2.12 True or False: Randomizing the source IP and port is a common defense against DNS spoofing.

○ True ● False

Solution: False. Randomizing the source IP would break the functionality of DNS, because you will not be able to receive the DNS response.

Randomizing the port is okay because you still receive the response. Your computer would just need to remember what random port it sent the request with and look for the answer at the same port.

Q2.13 True or False: Of the security principles covered in class, two factor authentication is best described as an example of defense in depth.

● True ○ False

Solution: True. 2FA means that even if one layer of your security is compromised (e.g. your password is stolen), other defenses remain in place to protect your account.

Q2.14 True or False: DNS (without DNSSEC) is secure against an on-path attacker, but not a MITM attacker.

○ True ● False

Solution: False. The on-path attacker can see the ID field and race the legitimate response.

Q2.15 True or False: ASLR prevents all buffer overflow attacks.

○ True ● False

Solution: False. ASLR can't prevent overwriting local variables.

Q2.16 True or False: Log analysis is effective at detecting attacks in real-time.

○ True ● False

Solution: False. Logs are usually checked offline afterwards, so they usually detect attacks after they've already happened.

Q2.17 True or False: Clickjacking can help an attacker execute reflected XSS attacks.

● True ○ False

Solution: True. Clickjacking can cause the victim to click on an attacker-crafted link, and reflected XSS requires the victim to click on an attacker-crafted link.

Q2.18 True or False: `EvanBot is a real bot.` (0 points)

● True ○ False

Solution: True. See http://isevanbotreal.com to learn more.

This is the end of Q2. Leave the remaining subparts of Q2 blank on Gradescope, if there are any. Proceed to Q3 on your answer sheet.

Q3 *Indirection* (24 points)

Consider the following vulnerable C code:

```
1  #include <stdlib.h>
2  #include <string.h>
3
4  struct log_entry {
5      char title[8];
6      char *msg;
7  };
8
9  void log_event(char *title, char *msg) {
10     size_t len = strnlen(msg, 256);
11     if (len == 256) return; /* Message too long. */
12     struct log_entry *entry = malloc(sizeof(struct log_entry));
13     entry->msg = malloc(256);
14     strcpy(entry->title, title);
15     strncpy(entry->msg, msg, len + 1);
16     add_to_log(entry); /* Implementation not shown. */
17 }
```

Assume you are on a little-endian 32-bit x86 system and no memory safety defenses are enabled.

Q3.1 (3 points) Which of the following lines contains a memory safety vulnerability?

- ◯ (A) Line 10
- ◯ (B) Line 13
- ● (C) Line 14
- ◯ (D) Line 15
- ◯ (E) —
- ◯ (F) —

Solution: Line 14 uses a `strcpy`, which is not a memory-safe function because it only terminates when it sees a null byte. The attacker could provide a string that is longer than the buffer or not properly null-terminated, and `strcpy` would still copy the entire string into the buffer, overwriting other variables in the process.

Note that line 15 uses a `strncpy` whose length parameter comes from `strnlen`, so it is safe.

Q3.2 (3 points) Seeing an opportunity to exploit this program, you fire up GDB and step into the `log_event` function. Give a GDB command that will show you the address of the rip of the `log_event` function. (Abbreviations are fine.)

Enter your answer in the text box on Gradescope.

◯ (G) — ◯ (H) — ◯ (I) — ◯ (J) — ◯ (K) — ◯ (L) —

Solution: `info frame` (abbreviated `i f`) would be the easiest command to use. Other solutions exist.

Q3.3 (3 points) Fill in the numbered blanks on the following stack and heap diagram for `log_event`. Assume that lower-numbered addresses start at the bottom of both diagrams.

Clarification during exam: The stack diagram shown is incorrect. The values 1 and `title` should should be swapped on the stack.

Stack

`title`
1
rip
sfp
`len`
`entry`

Heap

3
2

◯ (A) 1 = `entry->msg` 2 = `entry->title` 3 = `msg`

◯ (B) 1 = `entry->msg` 2 = `msg` 3 = `entry->title`

● (C) 1 = `msg` 2 = `entry->title` 3 = `entry->msg`

◯ (D) 1 = `msg` 2 = `entry->msg` 3 = `entry->title`

◯ (E) ——

◯ (F) ——

Solution: We messed up the stack diagram for this part. Arguments are pushed on the stack in reverse order, so 1 (`msg`) should actually be above `title`, not below title. Since it's our mistake, we are giving full credit to everyone on this subpart.

The two arguments, `title` and `msg`, must be on the stack, so 1 = `msg`.

Structs are filled from lower addresses to higher addresses, so 2 = `entry->title` and 3 = `entry->msg`.

Using GDB, you find that the address of the rip of `log_event` is `0xbfffe0f0`.

Let `SHELLCODE` be a 40-byte shellcode. Construct an input that would cause this program to execute shellcode. Write all your answers in Python 2 syntax (just like Project 1).

Q3.4 (6 points) Give the input for the `title` argument.

Enter your answer in the text box on Gradescope.

◯ (G) —— ◯ (H) —— ◯ (I) —— ◯ (J) —— ◯ (K) —— ◯ (L) ——

Solution: The `strcpy` at line 14 lets us write as much data as we want into `entry->title`, which is a buffer on the heap.

The `strncopy` at line 15 lets us write as much data as we want into `entry->msg`, which is a pointer on the heap. Note from the stack diagram that `entry->msg` is directly above `entry->title`, which is a buffer that we can overflow! Thus we can also overflow the `entry->msg` pointer and make it point wherever we want our data to be written.

The overarching idea is to use line 14 to overflow the `entry->msg` pointer to point to the rip, and then use line 15 to overwrite the rip (which `entry->msg` is now pointing to) with the classic buffer overflow exploit.

The `title` input should overflow `entry->msg` with the rip. We write 8 bytes of garbage to overwrite `entry->title`, then overwrite `entry->msg` with the rip.

```
'A' * 8 + '\xf0\xe0\xff\xbf'
```

Q3.5 (6 points) Give the input for the `msg` argument.

Enter your answer in the text box on Gradescope.

○ (A) —— ○ (B) —— ○ (C) —— ○ (D) —— ○ (E) —— ○ (F) ——

Solution: Now that `entry->msg` is pointing at the rip, our input for `msg` will be directly written to the rip (thanks to line 15).

Thus we can write the classic buffer overflow exploit (project 1, question 1): we overwrite the rip with the address directly above it (rip + 4), then write the shellcode. This causes the rip to point to shellcode.

```
'\xf4\xe0\xff\xbf' + SHELLCODE
```

Q3.6 (3 points) Which of the following defenses on their own would prevent your exploit?

Note: If stack canaries are enabled, you can assume `0xbfffe0f0` is still the correct address of the RIP.

☐ (G) Stack canaries

■ (H) W^X

■ (I) ASLR

☐ (J) None of the above

☐ (K) ——

☐ (L) ——

Solution: Stack canaries would not defend against this attack because we are not consecutively writing from the local variables to the rip. Instead, we are overflowing a heap variable (no canaries on the heap) and then directly writing above the canary.

W^X defends against your exploit by preventing the shellcode on the stack from being executed.

ASLR defends against your exploit by randomizing the address of the rip that you use in your exploit.

This is the end of Q3. Leave the remaining subparts of Q3 blank on Gradescope, if there are any. Proceed to Q4 on your answer sheet.

Q4 *Malcode* **(12 points)**

Q4.1 (3 points) Malcode X spreads by making a copy of its own binary on another machine and executing it. Which intrusion detection technique is best for detecting this malcode?

● (A) Signature-based detection
○ (B) Anomaly-based detection
○ (C) Specification-based detection
○ (D) Behavioral detection
○ (E) ——
○ (F) ——

Solution: Because the malcode does not change each time it replicates, we can add a signature for the malcode binary to detect and block it.

Q4.2 (3 points) Malcode X connects to other machines using TLS. Which intrusion detection method is best for detecting this malcode?

Select one option, and briefly justify your answer (1 sentence) in the text box.

○ (G) NIDS ● (H) HIDS ○ (I) —— ○ (J) —— ○ (K) —— ○ (L) ——

Solution: Because TLS is encrypted, the NIDS does not have the necessary host context in order to decrypt and inspect the traffic for the malcode. Thus, only the HIDS can defend against the malcode.

We may consider accepting NIDS if you explain that you can give the NIDS the server's private keys and let it actively intercept every connection.

Q4.3 (3 points) Malcode Y spreads by encrypting its binary, copying the encrypted binary and a decryption script to another machine, and executing the decryption script to run the malcode. The encryption key and the IV/nonce (if needed) are randomly generated each time the malcode replicates. Which encryption schemes would cause every copy of the malcode to look different?

Clarification during exam: "Cause every copy of the malcode to look different" means that the encrypted copies of the malcode differ in at least 1 byte.

■ (A) AES-ECB
■ (B) AES-CBC
■ (C) AES-CTR
☐ (D) None of the above
☐ (E) ——
☐ (F) ——

Solution: In all of these AES ciphers, the ciphertext looks different as long as the key is different each time.

Note that AES-ECB is deterministic with the same key, but changing the key still causes the ciphertext to look different.

Q4.4 (3 points) Malcode Z spreads the same way as Malcode Y. However, instead of randomly generating the encryption key and the IV/nonce (if needed), they are hard-coded into the binary and the decryption script. Which encryption schemes would cause every copy of the malcode to look different?

☐ (G) AES-ECB ☐ (I) AES-CTR ☐ (K) ——

☐ (H) AES-CBC ■ (J) None of the above ☐ (L) ——

Solution: A static key and IV means that the encrypted payload always remains the same.

Note that AES-CBC and AES-CTR are both deterministic if you use the same key and IV/nonce every time.

This is the end of Q4. Leave the remaining subparts of Q4 blank on Gradescope, if there are any. Proceed to Q5 on your answer sheet.

Q5 *CalCentral Security* **(20 points)**

Given your performance as a skilled attacker of the UnicornBox website, university administrators have asked you to assess the security of the CalCentral platform.

The CalCentral website is set up as follows:

- CalCentral is located at `https://calcentral.berkeley.edu/`.
- The Central Authentication Service (CAS) is located at `https://auth.berkeley.edu/`.
- CalCentral uses session tokens stored in cookies for authentication, similar to Project 3. The session token cookie has domain `berkeley.edu`, and the `Secure` and `HttpOnly` flags are set.
- CalCentral does **not** use CSRF tokens or any form of CSRF protection.

Each subpart is independent.

Q5.1 (3 points) When a user attempts to sign in on CalCentral, the CAS login portal appears in a pop-up window.

TRUE OR FALSE: Because CalCentral and CAS have the same origin, CAS can update the CalCentral webpage when a user signs in successfully.

◯ (A) True, because CalCentral and CAS are managed by the same organization.

◯ (B) True, because windows with the same origin can interact with each other.

◯ (C) False, because pop-up windows can never affect other windows, regardless of the origin.

● (D) False, because CalCentral and CAS don't have the same origin.

◯ (E) ——

◯ (F) ——

Solution: False. These pages might be able to communicate in other ways, but they have different origins under the same-origin policy.

Q5.2 (3 points) When a user attempts to sign in on CalCentral, the CAS login portal appears in an `iframe` embedded on the CalCentral page.

TRUE OR FALSE: This design allows CalCentral to modify the text field on the CAS website to autofill the username field.

◯ (G) True, because CalCentral and CAS are managed by the same organization.

◯ (H) True, because the inner frame is loaded with the same origin of the outer frame.

◯ (I) False, because Javascript is needed to autofill form fields.

● (J) False, because the outer frame cannot affect the contents of the inner frame.

◯ (K) ——

◯ (L) ——

Solution: False. Frame isolation states that the outer page cannot change the contents of the inner page, and inner pages cannot change the content of the outer page.

Q5.3 (3 points) If a user is logged into CalCentral (has a valid session token cookie), a GET Request to `https://calcentral.berkeley.edu/api/photo/` will contain a response with their CalCentral photo. The website `https://evil.com/` loads an image with the following HTML snippet:

```
<image src="https://calcentral.berkeley.edu/api/photo/">
```

TRUE OR FALSE: If a user is currently signed into CalCentral, the `https://evil.com/` website will be able to successfully display their photo.

● (A) True, because the browser attaches the session token in the request to CalCentral.

◯ (B) True, because the referer in the request is `https://calcentral.berkeley.edu`.

◯ (C) False, because the browser does not attach the session token in the request to CalCentral.

◯ (D) False, because the referer in the request is `https://evil.com`.

◯ (E) ——

◯ (F) ——

Solution: True. The browser will attach the user's session cookie for CalCentral, due to cookie policy (domain matches). Because CSRF protection is disabled, the server doesn't check for a CSRF token or validate the referer, so any requests with a valid session token will be sent back the appropriate profile picture.

Q5.4 (3 points) You find a reflected XSS vulnerability on CAS. `https://berkeley.edu` has a footnote that says "UC Berkeley."

TRUE OR FALSE: Using this vulnerability, you can cause the victim to see "CS 161 Enterprises" in the footnote when they visit `https://berkeley.edu`.

Clarification during exam: The footnote on `https://berkeley.edu/` is part of the static HTML page.

◯ (G) True, because the script runs with the same origin as `https://berkeley.edu`.

◯ (H) True, because XSS subverts the same-origin policy.

● (I) False, because the script runs with a different origin from `https://berkeley.edu`.

◯ (J) False, because the script only affects the browser's local copy of the site.

◯ (K) ——

○ (L) ——

Solution: False. Even with a reflected XSS vulnerability, all injected scripts would run with the origin of CAS (`https://auth.berkeley.edu`, which is different from the origin of `https://berkeley.edu/`. Thus, according to the the same-origin policy, CAS wouldn't be able to modify the `https://berkeley.edu` site.

Q5.5 (3 points) You find a stored XSS vulnerability on CalCentral.

TRUE OR FALSE: Using this vulnerability, you can cause the victim to load CalCentral with the "My Academics" button changed to link to `https://evil.com/`.

● (A) True, because Javascript on a page can change that page's HTML

○ (B) True, because CalCentral does not implement CSRF tokens.

○ (C) False, because Javascript on a page cannot change that page's HTML

○ (D) False, because `https://evil.com` has a different origin from CalCentral

○ (E) ——

○ (F) ——

Solution: True. A stored XSS vulnerability on CalCentral would allow an attacker to modify any of the contents of the CalCentral page.

Q5.6 (5 points) When a GET Request is made to `https://calcentral.berkeley.edu/api/classes/`, the server checks if the request's cookies contain the user's CalCentral session token. If a valid session cookie is found, the response contains a list of that user's classes. Otherwise, the server responds with `Error`.

Assume that you control each of the domains below. Select all domains where you'd be able to retrieve the class list of a victim who's currently signed in to CalCentral.

Hint: Recall that CalCentral does not use CSRF tokens or any form of CSRF protection.

Clarification during exam: "Retrieve the class list" means that the attacker is able to learn the class list.

☐ (G) `https://evil.edu/`

☐ (H) `https://berkeley.edu/`

☐ (I) `https://auth.berkeley.edu/`

☐ (J) `https://evil.calcentral.berkeley.edu/`

☐ (K) `http://calcentral.berkeley.edu/`

■ (L) None of the above

Solution:

Update, December 19, 2020 7:00pm PT: We dropped this question and gave everyone full points.

The original intended solution was all of the above. Because CSRF protection is disabled, all outbound requests made to `https://calcentral.berkeley.edu/api/classes/` will contain the session cookie. Since the attacker controls the website, they can add some malicious Javascript to read the response from the request and send the class data to the attacker.

This is the end of Q5. Leave the remaining subparts of Q5 blank on Gradescope, if there are any. Proceed to Q6 on your answer sheet.

Q6 *To Believe or Not To Believe* **(15 points)**

You are a detective at the Universal Conflict-resolution Bureau (UCB). You have been presented with a new case: Alice claims that Bob agreed to pay her $100. As evidence, she has a message from Bob, "I, Bob, owe Alice $100," along with some cryptography applied to the message.

Decide whether each piece of cryptographic evidence below is sufficient to believe her claim that this message is from Bob.

- m is the message from Bob.
- PK, SK is a public-private key pair.
- MAC is a cryptographically secure message authentication code function.
- k_1 and k_2 is a secret key shared between Alice and Bob.

  Clarification during exam: k_1 and k_2 are both secret keys shared between Alice and Bob.
- H is a cryptographically secure hash function.
- $\mathsf{Sign}(\mathsf{SK}, m)$ is a digital signature algorithm signing a message m with secret key SK.
- $\mathsf{Enc}, \mathsf{Dec}$ is an IND-CPA secure symmetric encryption scheme.

Clarification during exam: For all parts, you verify any signature you are presented with against the corresponding public key.

Clarification during exam: For all parts, Alice is not capable of stealing other people's private keys.

Clarification during exam: All subparts are independent.

Q6.1 (3 points) Alice presents you with $\mathsf{Sign}(\mathsf{SK}, m)$ and PK.

You obtain $\mathsf{Sign}(\mathsf{SK}_{CA}, \text{"Bob's public key is PK"})$ from a certificate authority you trust. SK_{CA} is the secret key of the CA, and you know the corresponding public key.

- (●) (A) m must be from Bob.
- ( ) (B) m is not necessarily from Bob.
- ( ) (C) ——
- ( ) (D) ——
- ( ) (E) ——
- ( ) (F) ——

Solution: Since you trust the certificate authority, you trust that the public key PK actually belongs to Bob. Then you can use Bob's public key to verify that the signature is valid.

Q6.2 (3 points) Alice presents you with $\mathsf{H}(m)$.

- ( ) (G) m must be from Bob.
- (●) (H) m is not necessarily from Bob.
- ( ) (I) ——
- ( ) (J) ——
- ( ) (K) ——
- ( ) (L) ——

Solution: Alice can compute the hash herself on any message.

Q6.3 (3 points) Alice presents you with $\mathsf{MAC}(k_1, m)$ and the secret key k_1.

◯ (A) m must be from Bob.

● (B) m is not necessarily from Bob.

◯ (C) ——

◯ (D) ——

◯ (E) ——

◯ (F) ——

Solution: Alice can compute a MAC on any message, since she has possession of the secret key k_1.

Q6.4 (3 points) Alice presents you with $\mathsf{MAC}(k_1, \mathsf{Enc}(k_2, m))$ and the secret keys k_1 and k_2.

◯ (G) m must be from Bob.

● (H) m is not necessarily from Bob.

◯ (I) ——

◯ (J) ——

◯ (K) ——

◯ (L) ——

Solution: Alice can compute the MAC and the encryption since she has the secret key k_1.

Q6.5 (3 points) Alice presents you with $\mathsf{Sign}(\mathsf{SK}, m)$, PK.

Additionally, Alice generates a certificate with her public key, $\mathsf{Sign}(\mathsf{SK}_{\text{Alice}}$, "Bob's public key is PK") and presents you with the certificate and her public key $\mathsf{PK}_{\text{Alice}}$.

Clarification during exam: Alice generates a certificate with her private key, not her public key.

◯ (A) m must be from Bob.

● (B) m is not necessarily from Bob.

◯ (C) ——

◯ (D) ——

◯ (E) ——

◯ (F) ——

Solution: Alice can be lying about Bob's public key PK.

This is the end of Q6. Leave the remaining subparts of Q6 blank on Gradescope, if there are any. Proceed to Q7 on your answer sheet.

Q7 *DKIM* (20 points)

DKIM (DomainKeys Identified Mail) is an email integrity scheme that prevents an attacker from forging the sender (`From:`) field and thus spoofing an email.

In DKIM, each mail server has a domain, such as `1e100.net`[1], and many public-private key pairs for signing emails. Each key pair has a unique numerical key ID.

To send an email, the sending mail server first verifies the sender's identity. Then the mail server attaches a DKIM header to the email. The header contains a signature on the email, the server's domain, and the ID of the key used to generate the signature.

To verify a DKIM header, you need to look up a public key to verify the signature. The public keys are stored using DNSSEC. To find a DKIM public key with domain `{DOMAIN}` and key ID `{KEY_ID}`, you make a DNSSEC query for `{Key_ID}._domainkey.{DOMAIN}` and receive a `TXT` type record with the public key.

For example, the name server for `20161025._domainkey.1e100.net` has a `TXT` record with a public key. The public key has ID `20161025` and is used by the `1e100.net` mail server.

Clarification during exam: Assume that `_domainkey` subdomain directly issues TXT records for `{KEY_ID}._domainkey`. For example, the `_domainkey.1e100.net` nameserver directly issues TXT records for `20161025._domainkey.1e100.net`.

Q7.1 (3 points) Which attacks on their own would help you learn how many DKIM keys a mail server is using?

- ■ (A) Zone enumeration with NSEC
- ■ (B) Zone enumeration with NSEC3
- ☐ (C) Kaminsky attack
- ☐ (D) None of the above
- ☐ (E) ——
- ☐ (F) ——

Solution: DNSSEC includes either NSEC or NSEC3, which both enable you to enumerate the subdomains present under `_domainkey.1e100.net`. Each time you query for a non-existent domain, you receive an NSEC record saying "No domains exist between `{PREVIOUS_KEY}._domainkey.1e100.net` and `{NEXT_KEY}._domainkey.1e100.net`, which lets you learn some keys that exist. If you make enough non-existent queries to traverse the entire alphabet, you've learned all the DKIM keys that exist under that mail server.

In NSEC3, you would need to reverse the hashed domains returned in the NSEC3 records, but as shown in lecture, this is feasible with modern computing power.

The Kaminsky attack does not help you learn about all the subdomains that exist under a given domain.

Consider making a DNSSEC query to obtain the public key with ID `12875102` for the domain `stanfraud.com`. Assume that you start with an empty cache (except the hardcoded KSK of the root). Assume that the `_domainkey` subdomain uses a different KSK and ZSK from its parent domain.

[1] This is a Google domain that Google uses for various infrastructure. Yes, Google gets to spy on all your Berkeley email.

For each of the next 3 subparts, list the records of each type in your cache after the query. Your answer should be a list of domains, e.g. `google.com`, `.` (root), `org`, `_domainkey.stanfraud.com`.

Q7.2 (3 points) `DS` type record(s) with hash(es) of the KSKs of the following name server(s):

Enter your answer in the text box on Gradescope.

◯ (G) —— ◯ (H) —— ◯ (I) —— ◯ (J) —— ◯ (K) —— ◯ (L) ——

Solution: `com`, `stanfraud.com`, `_domainkey.stanfraud.com`

The root returns a DS record to endorse the `com` name server. Then `com` returns a DS record to endorse the `stanfraud.com` name server. Finally, the `stanfraud.com` name server returns a DS record to endorse the `_domainkey.stanfraud.com` name server.

Q7.3 (3 points) `DNSKEY` type record(s) with the public key(s) (either ZSK or KSK) of the following name server(s):

Enter your answer in the text box on Gradescope.

◯ (A) —— ◯ (B) —— ◯ (C) —— ◯ (D) —— ◯ (E) —— ◯ (F) ——

Solution: `.` (root), `com`, `stanfraud.com`, `_domainkey.stanfraud.com`

Each of the name servers you contact must return their own public keys so that you can verify their signatures.

Q7.4 (3 points) `TXT` type record(s) with the public key(s) of the following name server(s) or mail server(s):

Enter your answer in the text box on Gradescope.

◯ (G) —— ◯ (H) —— ◯ (I) —— ◯ (J) —— ◯ (K) —— ◯ (L) ——

Solution: `12875102._domainkey.stanfraud.com`

As described in the question, the `TXT` record contains the final DKIM key with a given ID and a given domain. None of the other name servers need to return `TXT` answers, since they are performing standard DNSSEC.

Q7.5 (5 points) Which of the following private keys, if stolen on their own, would let an attacker create DKIM signatures as the `1e100.net` mail server? Select all that apply.

Clarification during exam: The DKIM key refers to the private key in the DKIM key pair.

■ (A) ZSK for the `net` name server

■ (B) KSK for the `1e100.net` name server

☐ (C) KSK for the `web.1e100.net` name server

■ (D) A DKIM key for the `1e100.net` domain

■ (E) ZSK for the `_domainkey.1e100.net` domain

☐ (F) None of the above

Solution: Any private signing key along the chain of trust from the root to `{KEY_ID}._domainkey.1e100.net` will help the attacker sign a fake `TXT` record with the attacker's public key. Then the attacker can use their own private key to create DKIM signatures.

For example, stealing the ZSK of the `.net` name server lets you sign a DS record endorsing the `1e100.net` name server. Since you've stolen the ZSK, you can sign a fake DS record that claims to endorse the `1e100.net` name server but is actually endorsing the attacker's public key. This process repeats until the attacker has claimed to endorse `_domainkey.1e100.net` name server (but is actually endorsing the attacker's public key). Then the attacker can use their own private key to create DKIM signatures.

Stealing the DKIM key directly also lets the attacker create DKIM signatures.

Q7.6 (3 points) Consider 2 versions of DKIM. In Version A, mail servers use the same key pairs indefinitely. In Version B, each week, every mail server switches all of its key pairs and publishes the old private keys.

You send an email with a valid DKIM signature, and someone else publishes that email months later. Which version(s) of DKIM would let you credibly deny that you sent that email?

Assume that nobody compromises your email account and nobody steals any private signing keys in DNSSEC and DKIM.

Clarification during exam: For Version B of DKIM, assume that the mail server publishes old public keys along with the old private keys.

◯ (G) Neither version

◯ (H) Version A only

● (I) Version B only

◯ (J) Both Version A and Version B

◯ (K) ——

◯ (L) ——

Solution: Version A: No, because anyone can make a DNSSEC query to learn the public key and verify the signature. Since the private key hasn't been published or stolen, the signature must have been made by the mail server, which authenticated you when attaching the signature. Thus the email must have been sent by you, and you cannot credibly deny that you sent it.

Version B: Yes, because once the mail server publishes the old private key, anyone can forge DKIM signatures. You can claim that you made this signature on a fake email with the published

private key, and there would be no way to tell whether the signature came from you or the mail server.

This is the end of Q7. Leave the remaining subparts of Q7 blank on Gradescope, if there are any. Proceed to Q8 on your answer sheet.

Q8 *ACK Flood* (12 points)

Recall that a SYN flooding attack overwhelms a server by sending a huge number of TCP packets with the SYN flag set.

ACK flooding is another DoS attack based on TCP, where the attacker sends a huge number of packets with the ACK flag set. This causes the server to exhaust resources trying to match each ACK packet to an existing TCP connection.

Q8.1 (4 points) Which of the following can be overwhelmed by an ACK flooding attack? Select all that apply.

☐ (A) Intermediate layer 3 routers (also called autonomous systems)

☐ (B) DNS name servers that only respond to DNS queries

■ (C) HTTP web servers that only respond to HTTP requests

■ (D) HTTPS web servers that only respond to HTTP and HTTPS requests

☐ (E) None of the above

☐ (F) ——

Solution: Intermediate routers pass along packets but don't process their contents, so they are not susceptible to ACK flooding. Also, the flood of ACK packets could take different paths through the network, and most networks implement some load balancing to avoid overwhelming any one router.

As shown in lecture, DNS name servers use UDP, so they would never process TCP packets.

HTTP and HTTPS both rely on TCP, so web servers answering HTTP requests are susceptible to ACK flooding.

Q8.2 (3 points) Which of the following could effectively defend against ACK flooding attacks while allowing legitimate connections? Select all that apply.

Hint: Recall that the ACK flag is used both in the TCP handshake and in sending messages over TCP.

☐ (G) Logging

☐ (H) NIDS

☐ (I) HIDS

■ (J) None of the above

☐ (K) ——

☐ (L) ——

Solution: HIDS reconstructs TCP connections, so it would be overwhelmed by ACK flooding attacks.

A stateful firewall/NIDS would reconstruct TCP connections and be overwhelmed by ACK flooding. A stateless firewall/NIDS could try dropping all TCP packets with the ACK flag set, but this would break legitimate connections communicating over TCP.

In reality, the only way to defend against such an attack is to have a lot of compute power. Companies like Cloudflare provide DoS protection services by maintaining a large amount of compute power, and routing its customers' requests through Cloudflare before forwarding them to its customers. If someone tries to ACK flood, it will be detected and stopped by Cloudflare, and only legitimate traffic reaches the actual server.

Q8.3 (5 points) Which field(s) in the packet headers must be set to a fixed value (cannot be random garbage) to target a specific process on a specific server?

- ☐ (A) Source IP
- ■ (B) Destination IP
- ☐ (C) Source port
- ■ (D) Destination port
- ☐ (E) Sequence number
- ☐ (F) None of the above

Solution: The destination IP must be fixed to target one server, and the destination port must be fixed to target one process on that server. All other fields can be random garbage.

This is the end of Q8. Leave the remaining subparts of Q8 blank on Gradescope, if there are any. Proceed to Q9 on your answer sheet.

Q9 *Pre-Master Secrets* **(15 points)**

Recall that in TLS, the client random value R_B, the server random value R_S, and the pre-master secret PS are used to derive the symmetric keys. This question considers several possible algorithms for deriving the symmetric keys.

For simplicity, suppose that we always generate one master key from R_B, R_S, and PS, and then we use a hash-based key derivation function (HKDF) to generate the four symmetric keys from the one master key.

Assume that R_B, R_S, and PS are all randomly generated integers between 0 and 2^{256} unless otherwise stated. For each master key derivation scheme below, select all attacks a man-in-the-middle can do.

Clarification during exam: The attacker learns the value of the master key as long as the attacker knows the key derived by the client and the key derived by the server.

Clarification during exam: For all parts, assume that the MITM has observed previous handshakes between the client and the server.

Clarification during exam: Assume that all addition operations are performed on unsigned, 256-bit integers. The output is taken modulo 2^{256}.

Clarification during exam: It is okay for the TLS handshake to fail, as long as the client/server derived a previously-used master key.

Q9.1 (3 points) Standard RSA TLS is used. To generate the master key, ignore PS, and add the other two values together: Master key $= R_B + R_S$.

■ (A) Learn the value of the master key

☐ (B) Make the server derive a previously-used master key in a future handshake

■ (C) Make the client derive a previously-used master key in a future handshake

☐ (D) None of the above

☐ (E) ——

☐ (F) ——

Solution: R_B and R_S are both sent in plaintext, so the MITM can learn the master key.

Because the client initiates the handshake, the MITM can force the client to derive any key MS by viewing R_B and sending back $R'_S = MS - R_B$.

This attack can't be executed on the server, since the server chooses R_S after the client chooses R_B. The MITM could try to send some malicious R'_B to the server, but they won't know what the server chooses as R_S until after they send R'_B, so they cannot cause the server to derive a previous master key.

Q9.2 (3 points) Standard RSA TLS is used. To generate the master key, ignore R_B, and add the other two values together: Master key $= R_S + PS$.

☐ (G) Learn the value of the master key

☐ (H) Make the server derive a previously-used master key in a future handshake

☐ (I) Make the client derive a previously-used master key in a future handshake

■ (J) None of the above

☐ (K) ——

☐ (L) ——

Solution: PS is sent encrypted, so the attacker cannot learn the master key.

If the attacker tries to replay an old handshake to the client, the client will choose a different PS, so the master key will be different.

If the attacker tries to replay an old handshake to the server, the server will choose a different R_S, so the master key will be different.

Q9.3 (3 points) Standard RSA TLS is used. To generate the master key, add the three values together: Master key $= R_B + R_S + PS$.

☐ (A) Learn the value of the master key

☐ (B) Make the server derive a previously-used master key in a future handshake

☐ (C) Make the client derive a previously-used master key in a future handshake

■ (D) None of the above

☐ (E) ——

☐ (F) ——

Solution: PS is sent encrypted, so the attacker cannot learn the master key.

R_B and R_S are different each time, so an attacker can't replay an old handshake in either direction. The client chooses $\{PS\}_{PK}$ randomly after receiving R_S, so the attacker can't exploit the properties of addition similar to the replay attack in other parts.

Q9.4 (3 points) A buggy version of RSA TLS is used. Instead of generating R_S randomly, the server increments R_S by 1 for every new connection. To generate the master key, add the three values together: Master key $= R_B + R_S + PS$.

☐ (G) Learn the value of the master key

■ (H) Make the server derive a previously-used master key in a future handshake

☐ (I) Make the client derive a previously-used master key in a future handshake

☐ (J) None of the above

☐ (K) ——

☐ (L) ——

Solution: PS is sent encrypted, so the attacker cannot learn the master key.

Suppose a previous connection used R_B, R_S, and PS. An attacker can initiate a handshake with the server and cause a previous master key to be derived by sending $R_B - 1$ to the server. Since the server will choose $R_S + 1$, the resulting master key is $(R_B - 1) + (R_S + 1) + PS = R_B + R_S + PS$, which is the same as the previous master key.

Q9.5 (3 points) A buggy version of RSA TLS is used. Instead of generating R_S randomly, the server increments R_S by 1 for every new connection. To generate the master key, hash the concatenation of the three values: Master key $= H(R_B||R_S||PS)$.

☐ (A) Learn the value of the master key

☐ (B) Make the server derive a previously-used master key in a future handshake

☐ (C) Make the client derive a previously-used master key in a future handshake

■ (D) None of the above

☐ (E) ——

☐ (F) ——

Solution: PS is sent encrypted, so the attacker cannot learn the master key.

The replay attack to the server from the previous part is no longer possible, because as long as the server sends back a different R_S, even if it's just $R_S + 1$, the master key will be different and unpredictable, no matter what R_B the attacker chooses.

This is the end of Q9. Leave the remaining subparts of Q9 blank on Gradescope, if there are any. Proceed to Q10 on your answer sheet.

Q10 *Is EvanBot Real?* (22 points)

The website `isevanbotreal.com` tells users whether EvanBot is real. However, the website creator doesn't want EvanBot to see the answer, so they require users to fill out a CAPTCHA before showing the answer.

The web server stores a database of CAPTCHA images and their corresponding text, where each CAPTCHA is given a unique ID.

```
1 CREATE TABLE captchas (
2     id INTEGER UNIQUE,
3     answer TEXT
4 );
```

When a client makes a GET request to `isevanbotreal.com/new`, the server returns a random CAPTCHA image and a cookie containing the ID number of that image. To submit the CAPTCHA, the client makes a POST request to `isevanbotreal.com/submit` with the answer text and the cookie containing the ID number of the image.

Q10.1 (5 points) To verify a submitted CAPTCHA, the server runs the following SQL query, replacing `$id` and `$text` with the cookie value and submitted answer text, respectively:

```
SELECT * FROM captchas WHERE id = $id and answer = '$text'
```

If zero rows are returned, the server returns `Incorrect CAPTCHA`. If more than zero rows are returned, the server returns the answer (`Yes`).

Provide an input for `$text` that would cause the server to return `Yes`, regardless of what the actual CAPTCHA text says.

If needed, you can use `$id` or `$text` to represent the value of user input.

Enter your answer in the text box on Gradescope.

◯ (A) —— ◯ (B) —— ◯ (C) —— ◯ (D) —— ◯ (E) —— ◯ (F) ——

Solution: The simplest solution is to inject something to make the query return the entire table. Possible solutions include:

```
' OR 1=1 --
```

```
'; SELECT * FROM captchas --
```

Other solutions exist.

Q10.2 (5 points) Consider a modification to the verification process. If zero rows are returned, the server returns `Incorrect CAPTCHA`, as before. If exactly one row is returned, the server returns `Yes`. If more than one row is returned, the server returns `Server error`.

Provide an input for `$text` that would cause the server to return `Yes`, regardless of what the actual CAPTCHA text says.

If needed, you can use `$id` or `$text` to represent the value of user input.

Enter your answer in the text box on Gradescope.

◯ (G) —— ◯ (H) —— ◯ (I) —— ◯ (J) —— ◯ (K) —— ◯ (L) ——

Solution: The simplest solution is to inject something to make the query return the entire table, then use SQL's `LIMIT` operator to limit the response to exactly one row:

```
' OR 1=1 LIMIT 1 --
```

```
'; SELECT * FROM captchas LIMIT 1 --
```

If you didn't remember the `LIMIT` operator, you can also take advantage of the `UNIQUE` constraint on the `id` field and build a new query that selects the one row with a given ID:

```
'; SELECT * FROM captchas WHERE id = $id
```

Other solutions exist.

Q10.3 (3 points) You want to create a malicious link with domain `isevanbotreal.com`. Any user who requests a CAPTCHA and then clicks on your link will receive the answer `Yes`. Is it possible to create such a link?

● (A) Yes, by exploiting an XSS vulnerability (if one exists) on `isevanbotreal.com`.

◯ (B) Yes, the link is `isevanbotreal.com/submit`.

◯ (C) No, because links make GET requests, and submissions are made through POST requests.

◯ (D) No, because links cannot be used for SQL injection.

◯ (E) ——

◯ (F) ——

Solution: An XSS vulnerability runs Javascript for any user who clicks on it, and the Javascript can make a POST request with input that "solves" the CAPTCHA.

`isevanbotreal.com/submit` is not a sufficient link, because a user who clicks on this makes a GET request, but we need the user to make a POST request with some data to solve the CAPTCHA.

Q10.4 (3 points) Consider the following Javascript pseudocode:

```
1 // make a GET request for a new CAPTCHA
2 fetch('http://isevanbotreal.com/new');
3
4 injection = ... // a correct SQL injection exploit from subpart 2
5
6 // send a POST request with an input that always solves the
    CAPTCHA
```

```
response = fetch('http://isevanbotreal.com/submit',
                 {method : 'POST', body : injection});

// Display the response
alert(response);
```

Does this code successfully display the answer (`Yes`) without solving the CAPTCHA?

● (G) Yes, because the browser attaches the ID cookie in the POST request

○ (H) Yes, because the script uses HTTP, not HTTPS

○ (I) No, because SQL injection cannot work over Javascript

○ (J) No, because the ID cookie is not sent to the server in the POST request

○ (K) ——

○ (L) ——

Solution: This script automates your attack in subpart 2. First it makes a request for a new CAPTCHA, which adds an ID cookie in your browser. Then it makes a POST request with the SQL injection exploit as the body. The browser will automatically attach the ID cookie to this exploit. Finally the script saves the response and displays it with `alert`.

HTTP/HTTPS is unrelated to this attack because it happens entirely on the application layer and doesn't require the presence of a network attacker.

If you used an XSS vulnerability to inject this Javascript snippet, you would cause anyone who clicks on your link (reflected XSS) or opens your malicious page (stored XSS) to immediately solve a CAPTCHA without actually solving it legitimately.

Q10.5 (3 points) Which of these defenses would stop your exploit in subpart 2? Select all that apply.

■ (A) Parameterized SQL

☐ (B) Return the ID as a hidden form field instead of a cookie

☐ (C) Set the `HttpOnly` flag on the ID cookie

☐ (D) None of the above

☐ (E) ——

☐ (F) ——

Solution: Parameterized SQL stops all SQL injection attacks by pre-compiling the query so that user input cannot be treated as code.

If the ID was a hidden form field, the attacker could still read its value and send it to the server.

Setting the `HttpOnly` flag on the ID cookie would not stop the attack, because the attacker can still read the value of the cookie, and the browser would still send the cookie to the server.

Q10.6 (3 points) Suppose the website implements a defense so that your attack in subpart 2 is no longer possible. There are no other vulnerabilities, and the CAPTCHAs are not solvable by machines, so the only way to receive the answer `Yes` is for a human to legitimately solve a CAPTCHA. Can a non-human still find a way to solve the CAPTCHAs?

● (G) Yes, by using a CAPTCHA solving service

○ (H) Yes, by brute-forcing the CAPTCHA

○ (I) No, because CAPTCHA solving services are prohibitively expensive

○ (J) No, because CAPTCHAs always defend against non-human attackers

○ (K) ——

○ (L) ——

Solution: As seen in lecture, CAPTCHA solving services are very cheap (pennies per CAPTCHA) and readily available. CAPTCHAs usually cannot be brute-forced, because there are too many possible answers.

This is the end of Q10. Leave the remaining subparts of Q10 blank on Gradescope, if there are any. Proceed to Q11 on your answer sheet.

Q11 *Hackerman Visits the Voting Booth* **(21 points)**

Your sketchy friend Jared asks you to use your CS 161 skills to help him rig some sort of election. He hands you a business card with credentials for a Russian supercomputer.

Armed with massive computing power, you show up to the Caltopia polling center. It has a Wi-Fi network secured with standard WPA2-PSK.

Q11.1 (5 points) You observe a WPA 4-way handshake. Which values from the handshake are needed to perform a brute-force search for the Wi-Fi password? Select all that apply.

- ■ (A) ANonce
- ■ (B) SNonce
- ■ (C) The router's MAC address
- ■ (D) The client's MAC address
- ■ (E) The MICs
- ☐ (F) None of the above

Solution: In the WPA2 4-way handshake, the information dependency goes {SSID, password} $\rightarrow$ PSK + {ANonce, SNonce, Router MAC, Client MAC} $\rightarrow$ PTK $\rightarrow$ MIC. In other words, the SSID and password are used to derive the PSK. Then the PSK, the nonces, and the MAC addresses are used to derive the PTK. Finally, the PTK is used to generate the MIC (message integrity/authentication codes).

To generate a guess for the PTK, you guess a password. Then you use your guessed password and the SSID to generate a guessed PSK. Then you use the guessed PSK, the nonces, and the MAC addresses to guess the PTK. Finally, you generate a MIC with your guessed PTK and see if it matches the MICs in the observed handshake.

Thus we need the nonces, the MAC addresses, and the MICs to perform our brute-force attack.

Q11.2 (4 points) What can you do after successfully brute-forcing the Wi-Fi password? Select all that apply.

- ■ (G) Perform on-path network attacks against victims in the same Wi-Fi network
- ■ (H) Decrypt network traffic encrypted with the PTK of a user who joins the network after you
- ■ (I) Decrypt network traffic encrypted with the GTK
- ☐ (J) Decrypt TLS network traffic
- ☐ (K) None of the above
- ☐ (L) ——

Solution:

You are on the local network, so you can enable sniffing mode and see packets of other people in the same Wi-Fi network. This makes you an on-path attacker.

With your brute-force attack, you now know the Wi-Fi password. As mentioned above, you are an on-path attacker, so when someone joins the network after you, you can observe their handshake. You use the SSID and your brute-forced password to derive the PSK. You use the nonces and the MAC addresses (sent in plaintext during the handshake, so you know their values), along with the PSK to derive the PTK. Now you can decrypt the victim's messages with their PTK.

During the 4-way handshake, the access point sends the GTK encrypted with the PTK. You have the PTK, so you can decrypt the GTK and then use it to decrypt any network traffic encrypted with the GTK. Alternatively, since you know the password, you can just initiate a WPA2 connection yourself and get the GTK value from the access point.

TLS is end-to-end secure, so being an on-path attacker in the local network won't help you decrypt TLS traffic.

Q11.3 (3 points) Which defenses would stop your attack? Select all that apply.

Clarification during exam: Assume that the Russian supercomputer is able to brute-force the password in in roughly an hour.

☐ (A) Changing the Wi-Fi password every day

■ (B) Using WPA2-Enterprise

☐ (C) A modern NIDS system

☐ (D) None of the above

☐ (E) ——

☐ (F) ——

Solution: Changing the password each day is generally a poor solution to a low-entropy password. However, we did not specify how long the Russian supercomputer takes to brute-force a password until clarifications, so everyone gets points for this answer choice.

A NIDS system protects a local network from external attacks, but does not stop attacks from local attackers.

WPA2-Enterprise is a good defense against this attack, because the attacker wouldn't be able to authenticate itself to the third-party server.

You arrive at the New Blackwell City polling center. It also has a Wi-Fi network secured with standard WPA2-PSK.

You walk up to a poll worker, claim that you're a fellow poll worker, and ask for the Wi-Fi password. They write the password on a post-it note and give it to you.

Q11.4 (3 points) Which security principle is most closely related to your experience at this polling place?

◯ (G) Consider Shannon's maxim

◯ (H) Least privilege

◯ (I) Security is economics

● (J) Consider human factors

◯ (K) Defense in depth

◯ (L) Time of check to time of use

Solution: Passing around passwords on post-it notes is an example of considering human factors, as seen in lecture.

Polling places are temporary employers which employ many people. An overworked, underpaid employee who already has the WiFi password written down and doesn't have mastery of low-level network attacks is unlikely to be a good defense against a convincing imposter.

At the Campanile City polling center, you see a DHCP Discover message broadcast to everyone.

Assume your computer has IP address `10.10.10.142`, and the network's router and DHCP server have IP address `10.10.10.5`. Assume that there are no other machines on the network. Assume there are no reserved or private IP addresses.

You want to return a malicious DHCP Offer that would make you a MITM. What values of the assigned IP address and the gateway IP address could you use in your response?

Q11.5 (3 points) Assigned IP address:

Enter your answer in the text box on Gradescope.

◯ (A) —— ◯ (B) —— ◯ (C) —— ◯ (D) —— ◯ (E) —— ◯ (F) ——

Solution: Any IP address not already in use works here. Since there are no other machines on the network, and we are ignoring reserved or private IP addresses, any IP except `10.10.10.142` and `10.10.10.5` is correct.

Q11.6 (3 points) Gateway IP address:

Enter your answer in the text box on Gradescope.

◯ (G) —— ◯ (H) —— ◯ (I) —— ◯ (J) —— ◯ (K) —— ◯ (L) ——

Solution: You should make your own computer the gateway, so that the victim sends any outgoing messages to you first. Thus the only correct answer is `10.10.10.142` (your IP address).

This is the end of Q11. Leave the remaining subparts of Q11 blank on Gradescope, if there are any. You have reached the end of the exam.

C Function Definitions

```
size_t strnlen(const char *s, size_t maxlen);
```

The strnlen() function returns the number of characters in the string pointed to by s, excluding the terminating null byte ('\0'), but at most maxlen. In doing this, strnlen() looks only at the first maxlen characters in the string pointed to by s and never beyond s+maxlen.

```
char *strcpy(char *dest, const char *src);
```

The strcpy() function copies the string pointed to by src, including the terminating null byte ('\0'), to the buffer pointed to by dest. The strings may not overlap, and the destination string dest must be large enough to receive the copy.

```
char *strncpy(char *dest, const char *src, size_t n);
```

The strncpy() function is similar, except that at most n bytes of src are copied. Warning: If there is no null byte among the first n bytes of src, the string placed in dest will not be null-terminated.

If the length of src is less than n, strncpy() writes additional null bytes to dest to ensure that a total of n bytes are written.