# CS 161 Summer 2024 — Introduction to Computer Security — Final

Name: ______________________ SID (person to your left): ______________________

Student ID: ______________________ SID (person to your right): ______________________

This exam is 170 minutes long.

| Question: | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Total |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Points: | 0 | 9 | 8 | 14 | 14 | 12 | 6 | 13 | 12 | 12 | 100 |

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

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

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

- ■ You can select
- ■ multiple squares (completely filled)

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

**Pre-exam activity** (0 points):

**Evanbot has been in their feels lately. Circle which Inside Out 2 Evanbot you feel most today.**

## Q1 *Honor Code* (0 points)

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

Read the honor code above and sign your name: ______________________

## Q2 *True/False* (9 points)

Each true/false is worth 0.5 points.

Q2.1 In a `printf` call, a `%x` format string specifier is matched with the argument `0x16161616`, which is the address of the string `"Evanbot wants pancakes!"`.

True or False: The `%x` format string specifier will make `printf` output `16161616`.

○ True ○ False

Q2.2 True or False: All memory safety vulnerabilities only exist because C allows users to write out-of-bounds.

○ True ○ False

Q2.3 True or False: Implementations of theoretically secure cryptographic schemes can leak information about plaintext.

○ True ○ False

Q2.4 Consider the following scheme: A message $M$ is encrypted with key $K_1$ in AES-ECB mode. An HMAC over the ciphertext is created using key $K_2$. Then, both the ciphertext and the tag are outputted.

True or False: This scheme is IND-CPA secure.

○ True ○ False

Q2.5 True or False: Using an unbiased source of entropy as a seed guarantees that a PRNG is rollback resistant.

○ True ○ False

Q2.6 Alice and Bob want to communicate using El Gamal. Alice sends a message $M$ to Bob. Mallory does not know $M$.

True or False: Mallory can manipulate Alice's ciphertext such that Bob receives $161 \times M$.

○ True ○ False

Q2.7 Assume that Alice trusts Evanbot. Mallory sends Alice a copy of Bob's certificate, which Alice verifies is signed by Evanbot.

True or False: Alice can trust this certificate has not been tampered with.

○ True ○ False

Q2.8 True or False: `https://toon.cs161.org` and `https://toon.cs161.org:81` have the same origin.

○ True ○ False

Q2.9 The `evanbot.com` server has ensured that all SQL statements in external facing APIs have been parameterized.

True or False: This prevents all SQL injection attacks on `evanbot.com`.

○ True ○ False

Q2.10 Assume all users are logged into `bank.com`, and that Facebook doesn't properly check user inputs. Mallory makes a Facebook post with the following content:

```
<img src="https://www.bank.com/transfer?amount=100&to=Mallory">
```

True or False: This triggers a stored XSS attack on all users who load and view this post in their browser.

○ True ○ False

Q2.11 True or False: In order for a NIDS to read traffic encrypted over TLS, it needs access to all of the private keys used by computers in the network.

○ True ○ False

Q2.12 True or False: NSEC records can be computed offline, but NSEC3 "white lies" records must be computed online.

○ True ○ False

Q2.13 True or False: TCP guarantees end-to-end encryption.

○ True ○ False

Q2.14 True or False: A Layer 2 packet is included as the payload inside a Layer 3 packet.

○ True ○ False

Q2.15 Assume Bob is outside Alice's LAN. Alice wants to send a message to Bob.

True or False: Alice's computer will create a packet for this message where the Layer 2 packet is addressed to the gateway in Alice's LAN, while the Layer 3 packet is addressed to Bob.

○ True ○ False

Q2.16 Alice and Bob send encrypted messages through a Tor relay.

True or False: The design of Tor intentionally prevents an attacker with a full (global) view of the entire network to learn that Alice and Bob are talking.

○ True ○ False

Q2.17 True or False: Prior to Snowden, HTTPS was almost impossible to use due to the high cost of certificates.

○ True ○ False

Q2.18 True or False: The steps to execute an evict and reload Spectre exploit are as follows: Train, Flush, Call, Reload.

○ True ○ False

Q2.19 (0 points) True or False: `EvanBot is a real bot.`

○ True ○ False

## Q3 *Memory Safety: Secret Exfiltration* (8 points)

Consider the following vulnerable C code:

```c
void vulnerable() {
    char command[64];
    memset(command, 0, 64);

    fread(command, 64, 1, stdin);
    system(command);
    gets(command);
}
```

In this question, your goal is to execute a 64-byte `SHELLCODE` with high probability.

- You run GDB once, and find that the address of `command` is `0xffff1210`.
- The `system` C function takes in a **null-terminated string** and runs that string as a command in the terminal.
- The `curl` terminal command sends a GET request to a specified URL. Example: `curl www.google.com`. There must be exactly one space between `curl` and the URL.
- Mallory controls the webpage `www.mallory.com/log?msg=X`, where `X` is a URL parameter. If someone loads this webpage, the URL parameter gets sent to Mallory.

Q3.1 (2 points) For this subpart only, all memory safety defenses are disabled.

For this subpart only, we always provide the harmless command `"ls"` as input to the `fread` call.

Which of these inputs to the `gets` call would cause the program to execute shellcode? Select all that apply.

☐ `SHELLCODE + 'A'*4 + \x10\x12\xff\xff`

☐ `'A'*4 + SHELLCODE + \x14\x12\xff\xff`

☐ `'shellcode'`

☐ `'shellcode' + '\0' + 'A'*54 + \x10\x12\xff\xff`

☐ None of the above

For subparts Q3.2–Q3.5, **stack canaries** are enabled, but all other defenses are disabled.

Q3.2 (2 points) Provide an input to `fread` for leaking the canary.

If any non-null byte works, use `'A'`.

Q3.3 (1 point) Mallory receives a URL parameter at her webpage. What bytes should she slice to extract the canary?

○ `[0:4]` ○ `[60:64]`

○ `[35:39]` ○ `[64:68]`

Q3.4 (2 points) Regardless of your answers to the previous subparts, let `CANARY` be the leaked 4-byte value of the canary.

Provide an input to `gets` for executing shellcode.

If any non-null byte works, use `'A'`.

Q3.5 (1 point) Would your exploit work if the first byte of the canary (lowest byte in memory) was always a null byte?

○ Yes, with no modifications.

○ Yes, but Mallory has to slice out some different bytes.

○ No, because leaking the canary is now impossible.

○ No, because writing the canary back into memory is now impossible.

## Q4 *Memory Safety: A Walk Down Memory Lane* (14 points)

For Q4.1-Q4.2, let's review C calling convention as described in lecture.

```
 1 # begin prologue
 2 push _____
 3 mov %esp, _____
 4 sub 16, %esp
 5 # end prologue
 6 ...
 7 # at this point the return value is stored in %eax
 8 # begin epilogue
 9 mov %esp, _____
10 pop _____
11 ret
12 # end epilogue
```

*Clarification: The ordering of registers in the* `mov` *instructions on lines 3 and 9 should be flipped.*

Q4.1 (1 point) What goes in the blank on **line 3**?

Q4.2 (1 point) When the instruction on **line 10** is run, which registers are affected?

The rest of this question is independent of the previous parts.

```
 1 char* rememberMe(char* coreMemory, size_t n) {
 2     fread(coreMemory, 1, n, stdin);
 3     return coreMemory;
 4 }
 5
 6 void nostalgia(size_t* bingbong) {
 7     char coreMemory[128];
 8     rememberMe(coreMemory, *bingbong);
 9     return;
10 }
11
12 int main() {
13     size_t* bingbong = malloc(2*sizeof(size_t));
14     printf("0x%x", bingbong);
15     fread(bingbong, sizeof(size_t), 2, stdin);
16     nostalgia(bingbong);
17     return 0;
18 }
```

| **Stack at Line 9** |
|---|
| RIP of `main` |
| SFP of `main` |
| (1) |
| (2) |
| (3) |
| SFP of `nostalgia` |
| `coreMemory[128]` |

The function `main` is being run. Your goal is to execute a 100-byte `SHELLCODE`. Suppose that:

- ASLR is enabled. All other defenses are disabled.
- At the end of a function call, if a function returns a non-void value, the value is stored in `%eax`.
- `%eax` is not modified at any other time.
- When line 14 executes, you see that the program has printed `0x12345678`.
- The command `call *%eax` is given by the two bytes `0xffd0`.

Q4.3 (1 point) Suppose the program is paused right before line 9 is executed. Fill in the stack diagram.

- ○ (1) RIP of `nostalgia` (2) `size_t n` (3) `coreMemory[128]`
- ○ (1) RIP of `nostalgia` (2) `coreMemory[128]` (3) `size_t* bingbong`
- ○ (1) `size_t* bingbong` (2) `size_t* bingbong` (3) RIP of `nostalgia`

Q4.4 (2 points) What should the **first** 4 bytes of input to `fread` on line 15 be? Write your answer as a decimal value. If multiple values are possible, write the smallest possible value.

Q4.5 (2 points) What should the **last** 4 bytes of input to `fread` on line 15 be?

If any non-null byte works, use `'A'`.

The following code is reproduced for your convenience.

```
 1 char* rememberMe(char* coreMemory, size_t n) {
 2     fread(coreMemory, 1, n, stdin);
 3     return coreMemory;
 4 }
 5
 6 void nostalgia(size_t* bingbong) {
 7     char coreMemory[128];
 8     rememberMe(coreMemory, *bingbong);
 9     return;
10 }
11
12 int main() {
13     size_t* bingbong = malloc(2*sizeof(size_t));
14     printf("0x%x", bingbong);
15     fread(bingbong, sizeof(size_t), 2, stdin);
16     nostalgia(bingbong);
17     return 0;
18 }
```

Q4.6 (4 points) What should the input to `fread` on line 2 be?

If any non-null byte works, use `'A'`.

Q4.7 (2 points) Select all additional memory safety defenses that would successfully prevent shellcode from executing if the exact exploit from before was used. Assume they are enabled independent of each other.

☐ Stack canaries

☐ Non-executable pages

☐ PACs (on a 64-bit system, assuming the exploit was adjusted for 64-bit)

☐ None of the above

Q4.8 (1 point) Suppose ASLR was disabled, but non-executable pages were enabled. Would the given code be secure if the attacker knew the location of the standard C library in memory?

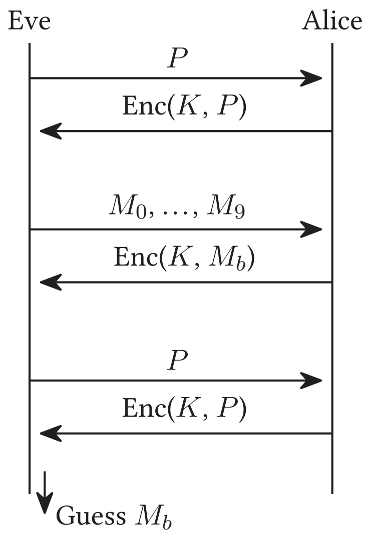
○ Yes ○ No

## Q5 *Cryptography: What Would Scheme Do* (14 points)

Evanbot wonders whether IND-CPA is truly the best way of assessing schemes, and wants to try some different games.

First, Evan creates IND-CPA10, a modification of IND-CPA that involves 10 messages instead of 2. The game works as follows:

1. Eve may send a polynomial number of plaintexts to Alice and recieve their ciphertexts.
2. Eve issues a **set of 10** distinct plaintexts of the same length, $(M_0, M_1, \ldots, M_9)$, to Alice.
3. Alice randomly chooses one of the messages $(M_b)$, encrypts it, and sends the encryption back to Eve.
4. Eve may continue to send any plaintexts to Alice to encrypt.
5. Eventually, Eve must guess what value Alice chose to encrypt.



Eve wins the game if she can guess $M_b$ with a probability greater than randomly guessing.

Q5.1 (1 point) Fill in the blank: In order for a scheme to be IND-CPA10 secure, Eve must be able to correctly guess $M_b$ with probability ________.

*Clarification during exam: The probability should be a numerical value, without comparators.*

Q5.2 (1 point) Select all encryption schemes that are IND-CPA10 secure.

☐ AES-CBC ☐ AES-CTR ☐ None of the above

☐ AES-ECB ☐ AES-CFB

Q5.3 (2 points) **For this subpart only**, assume that our IV/Nonce is set using a counter that starts at 0, and increments **every time Alice encrypts a message**.

Select all encryption schemes that remain IND-CPA10 secure.

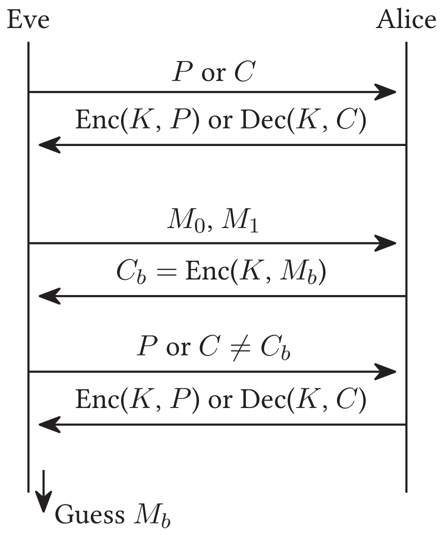
☐ AES-CBC ☐ AES-CFB

☐ AES-CTR ☐ None of the above

Evanbot isn't satisfied with IND-CPA10, and comes up with IND-BOT instead.

In this game, Eve can trick Alice into **both encrypting and decrypting** arbitrary messages. The game works as follows:

1. Eve may choose plaintexts to send to Alice and receive their ciphertexts. Eve may also choose ciphertexts to send and receive their plaintexts.
2. Eve issues a pair of distinct plaintexts of the same length, $M_0$ and $M_1$, to Alice.
3. Alice randomly chooses $M_b = M_0$ or $M_1$ to encrypt, and returns $C_b = \text{Enc}(K, M_b)$ to Eve.
4. Eve may continue to send plaintexts to encrypt, and she may send any ciphertexts that **are not** $C_b$ to decrypt.
5. Eventually, Eve must guess what value Alice chose to encrypt.



Eve wins the game if she can guess $M_b$ with probability greater than 0.5.

Evanbot sees that many common schemes are insecure against IND-BOT, and they want to know why.

*Clarification during exam: For Q5.4-5.6, assume the cipher's block size matches the message size (both constant), producing a one-block output for any input.*

Q5.4 (2 points) In two sentences or fewer, why is AES-ECB insecure against IND-BOT?

In each subpart, provide a sequence of events (choosing from the list below) to win the game.

A. Eve asks Alice to encrypt the plaintext ______.
Eve labels the returned ciphertext as ( ______ , ______ ).

B. Eve asks Alice to decrypt the ciphertext ( ______ , ______ ).
Eve labels the returned plaintext as ______.

C. Eve sends randomly-chosen challenge plaintexts $M_0, M_1$ and receives $(IV_b, C_b)$.

Write one event per row. You don't have to use all rows provided, but you may not use extra rows.

On each row: In the left box, write the letter (A to C) of the event. In the right box, write out the value(s) used in the placeholders, leaving spaces or commas to separate values.

In the final box: Using the variables we have, provide how Eve should guess which plaintext message

Alice encrypted. You may use mathematical operations in your answer.

A completely unrelated sample answer is provided, both in English and in exam box format:

1. Eve sends randomly-chosen challenge plaintexts $M_0, M_1$ and receives $(IV_b, C_b)$.
2. Eve asks Alice to encrypt the plaintext $M_0 \oplus 12$.
   Eve labels the returned ciphertext as $(IV_3, C_3)$.
3. Eve asks Alice to decrypt the ciphertext $(5, C_3)$.
   Eve labels the returned plaintext as $M_3$.

| | |
|---|---|
| C | |
| A | $M_0 \oplus 12$ $(IV_3, C_3)$ |
| B | $(5, C_3)$ $M_3$ |

If $IV_b \oplus IV_3 = M_3$, guess $M_1$. Otherwise, guess $M_0$.

Q5.5 (4 points) How is AES-CTR insecure against IND-BOT?

| | |
|---|---|
| | |
| | |
| | |
| | |

The event list and placeholders are repeated here for your convenience.

A. Eve asks Alice to encrypt the plaintext ______.
   Eve labels the returned ciphertext as ( ______ , ______ ).
B. Eve asks Alice to decrypt the ciphertext ( ______ , ______ ).
   Eve labels the returned plaintext as ______.
C. Eve sends randomly-chosen challenge plaintexts $M_0, M_1$ and receives $(IV_b, C_b)$.

Q5.6 (4 points) How is AES-CBC insecure against IND-BOT?

## Q6 *Cryptography: Yet Another Passwords Question (YAP Question)* (12 points)

EvanBot is maintaining a passwords database with $U$ users. Each user has a unique username. Each user chooses a password (not necessarily unique) from a pool of $P$ passwords.

Mallory is able to edit the database. Mallory wants to perform two attacks:

- Compute every user's password.
- Change every user's password to "potato," a password that is not in the passwords pool.

For each password storage scheme, select roughly how many hashes Mallory needs to compute for each attack.

**Scheme A**: For each user, store `username` and H(`password`).

Q6.1 (1 point) In Scheme A, how many hashes are needed to learn every user's password?

| | | |
|---|---|---|
| ○ $0$ | ○ $P$ | ○ $U+UP$ |
| ○ $1$ | ○ $2P$ | ○ $1+P$ |
| ○ $U$ | ○ $U+P$ | ○ $P+UP$ |
| ○ $2U$ | ○ $1+U$ | ○ $UP$ |

Q6.2 (1 point) In Scheme A, how many hashes are needed to change every user's password to "potato?"

| | | |
|---|---|---|
| ○ $0$ | ○ $P$ | ○ $U+UP$ |
| ○ $1$ | ○ $2P$ | ○ $1+P$ |
| ○ $U$ | ○ $U+P$ | ○ $P+UP$ |
| ○ $2U$ | ○ $1+U$ | ○ $UP$ |

**Scheme B**: For each user, store `username` and H(`username` || `password`).

Q6.3 (1 point) In Scheme B, how many hashes are needed to learn every user's password?

| | | |
|---|---|---|
| ○ $0$ | ○ $P$ | ○ $U+UP$ |
| ○ $1$ | ○ $2P$ | ○ $1+P$ |
| ○ $U$ | ○ $U+P$ | ○ $P+UP$ |
| ○ $2U$ | ○ $1+U$ | ○ $UP$ |

Q6.4 (1 point) In Scheme B, how many hashes are needed to change every user's password to "potato?"

| | | |
|---|---|---|
| ○ $0$ | ○ $P$ | ○ $U+UP$ |
| ○ $1$ | ○ $2P$ | ○ $1+P$ |
| ○ $U$ | ○ $U+P$ | ○ $P+UP$ |
| ○ $2U$ | ○ $1+U$ | ○ $UP$ |

**Scheme C**: For each user, store `username` and H(H(`username`) || `password`).

Q6.5 (2 points) In Scheme C, how many hashes are needed to learn every user's password?

| | | |
|---|---|---|
| ○ 0 | ○ $P$ | ○ $U + UP$ |
| ○ 1 | ○ $2P$ | ○ $1 + P$ |
| ○ $U$ | ○ $U + P$ | ○ $P + UP$ |
| ○ $2U$ | ○ $1 + U$ | ○ $UP$ |

Q6.6 (2 points) In Scheme C, how many hashes are needed to change every user's password to "potato?"

| | | |
|---|---|---|
| ○ 0 | ○ $P$ | ○ $U + UP$ |
| ○ 1 | ○ $2P$ | ○ $1 + P$ |
| ○ $U$ | ○ $U + P$ | ○ $P + UP$ |
| ○ $2U$ | ○ $1 + U$ | ○ $UP$ |

**Scheme D**: For each user, store `username` and H(`username` || H(`password`)).

Q6.7 (2 points) In Scheme D, how many hashes are needed to learn every user's password?

| | | |
|---|---|---|
| ○ 0 | ○ $P$ | ○ $U + UP$ |
| ○ 1 | ○ $2P$ | ○ $1 + P$ |
| ○ $U$ | ○ $U + P$ | ○ $P + UP$ |
| ○ $2U$ | ○ $1 + U$ | ○ $UP$ |

Q6.8 (2 points) In Scheme D, how many hashes are needed to change every user's password to "potato?"

| | | |
|---|---|---|
| ○ 0 | ○ $P$ | ○ $U + UP$ |
| ○ 1 | ○ $2P$ | ○ $1 + P$ |
| ○ $U$ | ○ $U + P$ | ○ $P + UP$ |
| ○ $2U$ | ○ $1 + U$ | ○ $UP$ |

## Q7 *Web Security: Don't Forget to Wash your Hands* (6 points)

Alice, a germaphobe, creates a search engine called `www.learninsideout.com` to rival the leading search engine which shall not be named.

When a user of `www.learninsideout.com` searches for a term $X$, they will be first redirected to a loading page that displays *Searching results for "$X$"....*

For example, if a user searches `"evanbot"` in the search bar, the loading page will display *Searching results for "evanbot"....*

Q7.1 (1 point) Which class of web attacks is this system vulnerable to?

- ○ SQL Injection
- ○ Clickjacking
- ○ CSRF
- ○ XSS

Q7.2 (1 point) Construct a search query that causes the user's browser to execute the Javascript `alert(1)`.

Alice sees the vulnerability, and as a lover of all things clean, is implementing input sanitization.

In the following subparts, we will explore multiple **broken** input sanitization schemes. For each subpart, provide a search query that executes the Javascript `alert(1)`. Each scheme is independent.

Q7.3 (1 point) Replace the last instance of </`script`> with `"none"`.

Q7.4 (1 point) Scan the input once and remove every instance of `alert` from the search query.

Q7.5 (1 point) Add `garbage` to the beginning (before the first character) of the search query.

Q7.6 (1 point) Name a broader defense that would prevent such attacks from occurring. Your answer should be 5 words or less.

## Q8 *Web Security: Doggo's Disaster Website* **(13 points)**

The newest hit social media network by developer Neta has brought the wonders of the internet to our furry friends through a website called Doggo! The URL for Doggo is `www.doggo.com`.

When a user creates a profile on `www.doggo.com`, they enter a `username` and a `password`. Each user has their own page that is publicly viewable at the URL, `www.doggo.com/view?user=H(username)`, where `H(username)` denotes the hashed value of `username`. If a user is logged in, they can edit the HTML of their own page.

Q8.1 (1 point) We discover that a secret is randomly generated and stored in a comment at `www.doggo.com/secret.html`. Can the user Alice steal the secret?

- ○ Yes, because the Same-Origin Policy says nothing about retrieving webpages.
- ○ Yes, because Alice's page and the secrets page have the same origin.
- ○ No, because Alice's page and the secrets page have different origins.
- ○ No, because the JS is being loaded from different webpages.

Q8.2 (1 point) Suppose Alice and Bob have accounts for their dogs with the corresponding URLs:

```
www.doggo.com/view?user=H(alice)
www.doggo.com/view?user=H(bob)
```

Alice configures some custom settings for her page through her browser cookies. If Bob uses the same computer and browser as Alice, will he see the customization when he logs in?

- ○ Yes, because the URLs have equal domain suffixes and different path prefixes.
- ○ Yes, because the URLs have equal domain suffixes and equal path prefixes.
- ○ No, because the URLs have equal domain suffixes and equal path prefixes.
- ○ No, all cookies are only applied to the exact website they were generated by.

The following subparts are **independent** from each other.

Consider the following modified scheme: When a user creates an account, a random 16-byte `user-id` is associated with the user. The URL for their profile is `www.doggo.com/view?user=user-id`.

The backend SQL table `dog` stores information for each user. Here are the attributes of the table:

| `dogs` Table | |
|---|---|
| `username` | `string` |
| `password` | `string` |
| `user-id` | `string` |
| `time-of-creation` | `int` |
| `is-active` | `boolean` |

The table attributes, reprinted:

`username` (`str`), `password` (`str`), `user-id` (`str`), `time-of-creation` (`int`), `is-active` (`boolean`).

*Clarification during exam: For the remainder of this question, the value of "username" stored in the dogs table is not hashed.*

Q8.3 (1 point) Write a SQL query that, given the username, `u1`, and the password, `p1`, returns the corresponding `user-id`.

Q8.4 (1 point) Mallory does not know Alice's `user-id`. Write a link that allows Mallory to view Alice's page.

Q8.5 (2 points) Alice is logged into `www.wallet.com`, which uses session-based authentication.

A GET request to `www.wallet.com/send?amt=100&to=bob` will transfer $100 to Bob.

Mallory wants to trick Alice into sending $100 to user `mallory`.

Fill in the blanks: Mallory can execute a $\underset{a}{\underline{\hspace{6em}}}$ attack by adding the HTML $\underset{b}{\underline{\hspace{6em}}}$ to her own page and having Alice go to $\underset{c}{\underline{\hspace{6em}}}$'s Doggo page.

Blank a:

Blank b:

Blank c:

Q8.6 (1 point) Select all valid defenses for the attack seen in Q8.5.

- ☐ SQL Input Sanitization
- ☐ Referer Validation
- ☐ CSRF Token
- ☐ None of the above

Neta's overzealous creator, Netabot, likes to view an updated list of their users daily by making a GET request to `www.doggo.com/list`. **Netabot is the only person who can access this page.** The list is formatted like this:

```
username, password, user-id, time-of-creation, is-active
evanbot, password123, 1, 2024-08-09, true
alice, password123, 2, 2024-08-09, false
...
```

`www.doggo.com/list` also has a "Delete All Users" button. The "Delete All Users" button makes a POST request to the endpoint `/deleteUsers`.

*Clarification during exam: The "Delete All Users" button makes a POST request to the endpoint www.doggo.com/deleteUsers.*

Q8.7 (1 point) Describe how Mallory can get Netabot to execute a simple `alert()` Javascript command. Assume no defenses are in place.

Q8.8 (1 point) Netabot catches on to this and employs input sanitization on the username. Describe how Mallory can cause all users to be deleted the next time Netabot accesses their list. Assume no other defenses are in place.

You may use the JavaScript function `post(URL)`, which sends a POST request to the given `URL`.

Netabot modifies the list page to show all images that users have stored on their webpages. When Netabot makes a GET request to `www.doggo.com/list`, Neta's server will run a compute-intensive scan on every image and blacklist any account with inappropriate images.

Q8.9 (1 point) What kind of attack could Mallory execute in order to overwhelm Neta and make it difficult to blacklist users? Your answer should be fewer than 5 words.

Q8.10 (1 point) Describe how Mallory can execute this attack. Your answer should be 1 sentence.

Q8.11 (1 point) Explain why this attack is possible despite Neta's server having more computational power than Mallory.

Q8.12 (1 point) What is a defense Neta can implement to slow down Mallory without imposing limits on users?

## Q9 *Networking: Don't Neglect Sadness* (12 points)

Joy has learned that Sadness is important to everyone's lives, and a valuable emotion to experience.

Joy wants to contact Sadness at `sadness.joy.cs161.org` to apologize and amend their relationship. Assume that Joy makes her request to a recursive resolver, which has cached the following records:

```
Record 1:   joy.cs161.org       A        192.195.66.0
Record 2:   joy.cs161.org       NS       joy-dns.cs161.org
Record 3:   a.org-servers.net   A        192.161.4.27
Record 4:   org                 NS       a.org-servers.net
```

Q9.1 (0.5 point) What does `joy.cs161.org` in `Record 1` represent?

○ Domain name ○ DNS Zone ○ IP address

Q9.2 (0.5 point) What does `joy.cs161.org` in `Record 2` represent?

○ Domain name ○ DNS Zone ○ IP address

Q9.3 (0.5 point) What does `joy-dns.cs161.org` in `Record 2` represent?

○ Domain name ○ DNS Zone ○ IP address

Q9.4 (0.5 point) What does `192.161.4.27` in `Record 3` represent?

○ Domain name ○ DNS Zone ○ IP address

Q9.5 (3 points) Mallory is an off-path attacker who wants to poison `sadness.joy.cs161.org`. She has tricked Joy into making DNS requests for the following ordered pairs of fake domains.

Assume each pair of fake domains is requested independently of each other (caching from the first domain sustains through the second, but not across the pairs).

Which of the following pairs of fake domains provide Mallory at least two attempts to poison the cache? Select all that apply.

☐ `fake161.berkeley.edu`, `fake162.berkeley.edu`

☐ `fake27.disgust.cs161.org`, `fake28.anger.cs161.org`

☐ `fake1.cs161.org`, `fake2.cs161.org`

☐ `fake61b.org`, `fake61c.org`

☐ `fake42.joy.cs161.org`, `fake43.joy.cs161.org`

☐ `fake6.ennui.com`, `fake7.envy.org`

For subparts Q9.6–9.7, assume DNS over Diffie-Hellman TLS is enabled. The recursive resolver's cache has not been modified (same four records from earlier).

Q9.6 (2 points) Joy wants to look up the IP address of `sadness.joy.cs161.org`. Which hosts will the recursive resolver need to make a TLS connection with in order to complete Joy's DNS over TLS request? Select all that apply.

- ☐ Joy's stub resolver
- ☐ `root` NS
- ☐ `.org` NS
- ☐ `cs161.org` NS
- ☐ `joy.cs161.org` NS
- ☐ `sadness.joy.cs161.org` NS

Q9.7 (2 points) Compared to DNS over UDP, what does DNS over TLS provide? Select all that apply.

- ☐ Defense against malicious name servers
- ☐ Confidentiality over the response
- ☐ Integrity over the response
- ☐ Defense against malicious recursive resolvers
- ☐ None of the above

Q9.8 (1 point) Fill in the blank, one word per blank: One of the core reasons DNSSEC is chosen instead of DNS over TLS is that DNSSEC provides $\underset{a}{\underline{\hspace{3cm}}}$ security while DNS over TLS provides $\underset{b}{\underline{\hspace{3cm}}}$ security.

Blank a:

Blank b:

Q9.9 (1 point) What is a benefit to Diffie-Hellman TLS that RSA TLS does not provide? Your answer should be fewer than 5 words.

Q9.10 (1 point) True or False: In Paul Vixie's lecture, one reason provided for creating DNS over HTTPS was to prevent attackers from detecting that a specific request was for DNS.

○ True ○ False

## Q10 *Networking: TCP-EVAN, coming soon to a network near you* (12 points)

Q10.1 (1 point) In TCP packets, why do we use sequence numbers?

- ○ To recover from corrupted packets.
- ○ To recover packets that arrive out of order.
- ○ To send packets to the correct process on the machine.
- ○ To verify which machine which packets should be sent to.

Q10.2 (1 point) Suppose the client and server use initial sequence numbers $A$ and $B$, respectively. After the handshake, the client sends the server some data of length $L$. When the server replies with one packet, what will the SEQ and ACK numbers of that packet be?

- ○ SEQ: $A + 1$ ACK: $B + L + 1$
- ○ SEQ: $B$ ACK: $L + 1$
- ○ SEQ: $B + 1$ ACK: $A + B + 1$
- ○ SEQ: $B + 1$ ACK: $A + L + 1$

Q10.3 (2 points) Which statements are true of TCP? Select all that apply.

- ☐ TCP allows for multiple connections between the same machines using port numbers.
- ☐ TCP guarantees that if a packet wasn't received, it will be resent until it is received.
- ☐ TCP guarantees the confidentiality of all packets sent in either direction.
- ☐ If you know the initial values used in the handshake, it is possible to determine how many bytes one side has sent using only one subtraction.
- ☐ None of the above

Evanbot devises a new scheme based on TCP called TCP-EVAN, which replaces SEQ and ACK numbers with a single 32-bit Element Valuation Number (EVN). A packet's EVN is the index of that packet in the overall stream of communication. For example:

1. Client and server complete the handshake using a random initial EVN, with the EVN of the final ACK packet of the handshake set to $C$.
2. The client sends the server one packet, and the EVN of that packet is $C + 1$.
3. The server replies with three packets, and the EVN of the last packet is $C + 4$.

Q10.4 (2 points) Which statements are true of TCP-EVAN? Select all that apply.

- ☐ TCP-EVAN allows for multiple connections between the same machines using port numbers.
- ☐ TCP-EVAN guarantees that if a packet wasn't received, it will be resent until it is received.
- ☐ TCP-EVAN guarantees the confidentiality of all packets sent in either direction.
- ☐ If you know the initial values used in the handshake, it is possible to determine how many packets one side has sent using only one subtraction.
- ☐ None of the above

For the following subparts, select whether or not TCP-EVAN could support the given capability. Explain why or why not.

Q10.5 (2 points) Given all packets sent across the connection, it is possible to split messages into two bytestreams: one per direction between client and server.

○ TCP-EVAN can support this already ○ TCP-EVAN cannot support this as is

Q10.6 (2 points) For both sides of the connection, it is possible to reorder packets that arrive out of order.

○ TCP-EVAN can support this already ○ TCP-EVAN cannot support this as is

Q10.7 (2 points) Blind hijacking by an off-path attacker is less likely to succeed than with TCP.

*Clarification during exam: Treat the left option as "True" and the right option as "False". Please still explain your answer.*

○ TCP-EVAN can support this already ○ TCP-EVAN cannot support this as is

## Post-Exam Activity

Nothing on this page will affect your grade.

Evanbot was not active on Ed this summer because Evanbot has been so busy recently. (`Bot apologizes for this. ~Evanbot`) What do you think they've been cooking?

## Comment Box

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