CS 161 Fall 2023 — Introduction to Computer Security — Final

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

PRINT your student ID: _____

Question:	1	2	3	4	5	6	7	8	9	10	11	Total
Points:	0	8	12	14	11	8	14	10	7	7	9	100

Pre-exam activity (for fun, not graded):

Try to guess the outcome of EvanBot's 8 coin flips! To play the game, fill in `H` or `T` in each of the 8 boxes. Statistically, a few students in the class will get all eight guesses correct! You could be that student...

To prove EvanBot has fairly flipped the coins, here's the hash of the commitment used to generate them:

```
41c9ef9c8752cabb33b6e762976cf8b777bea918082abfea2cc737a76de15180
```

Q1 *Honor Code* (0 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* **(8 points)**

Each true/false is worth 0.5 points.

Q2.1 EvanBot's website has a bug, where it crashes if the user chooses the username "pancakes." EvanBot is aware of the bug, but hopes that no one notices.

TRUE or FALSE: This is an example of defense in depth.

○ TRUE ○ FALSE

Q2.2 When you request a certificate from a certificate authority (CA), the CA must always compute a digital signature before sending you the certificate.

○ TRUE ○ FALSE

Q2.3 Consider a PRNG that is seeded once with a 128-bit value, and then used to generate 256 bits of output.

TRUE or FALSE: An attacker with infinite computational power can distinguish (with probability $> 50\%$) the output of the PRNG from 256 bits of truly random output.

○ TRUE ○ FALSE

Q2.4 TRUE or FALSE: It is possible for a domain to set a cookie that never gets sent back to that domain.

○ TRUE ○ FALSE

Q2.5 Consider a website that allows users to submit any arbitrary HTML to be displayed on the website, but prohibits users from submitting any JavaScript to be displayed on the website.

TRUE or FALSE: It is possible for an attacker to use this website to execute a CSRF attack.

○ TRUE ○ FALSE

Q2.6 TRUE or FALSE: Considering human factors is a security principle relevant to phishing attacks.

○ TRUE ○ FALSE

Q2.7 TRUE or FALSE: It is possible for browsers to implement a defense that completely stops clickjacking attacks.

○ TRUE ○ FALSE

Q2.8 CAPTCHAs stop automated attacks because the challenge can only be generated by a human.

○ TRUE ○ FALSE

Q2.9 A on-path attacker who records a WPA2 handshake can always learn the PSK without using brute force.

○ True ○ False

Q2.10 True or False: If an off-path attacker spoofs a TCP SYN packet to a server, the server can always detect that something has gone wrong and reply with a RST packet.

○ True ○ False

Q2.11 True or False: If an off-path attacker spoofs a TCP SYN packet to a server, the attacker will see the SYN-ACK response.

○ True ○ False

Q2.12 Firewalls are a reliable way to defend against ARP spoofing attacks.

○ True ○ False

Q2.13 Logging (the intrusion detection strategy) is an example of detecting if you can't prevent.

○ True ○ False

Q2.14 A behavioral detection system could use default-allow or default-deny policies.

○ True ○ False

Q2.15 Consider a piece of malware that outputs a copy of its code, and an HMAC on its code, using a randomly-generated key. Then, the malware sends a copy of the code and the HMAC to another computer.

True or False: This would be a reliable way to propagate while avoiding signature-based detection.

○ True ○ False

Q2.16 Consider a piece of malware that outputs an HMAC on its code, using a randomly-generated key. Then, the malware sends the HMAC to another computer.

True or False: This would be a reliable way to propagate while avoiding signature-based detection.

○ True ○ False

Q2.17 (0 points) True or False: `EvanBot is a real bot.`

○ True ○ False

Q3 *Memory Safety: exec* **(12 points)**

Consider the following C function:

```
1 void vulnerable() {
2     char command[16];
3     fread(command, 1, 24, stdin);
4     if (strcmp(command, "STOP") == 0) {
5         return;
6     }
7     exec(command);
8 }
```

The `exec(char* arg)` function replaces the currently running program with the program found at filename `arg`. This function does not return control back to the original caller function. You can assume that `exec` crashes if `arg` refers to a nonexistent file.

EvanBot runs GDB once and finds that the address of `command` is `0xffff1024`.

EvanBot's goal is to cause this function to run an 8-byte `SHELLCODE`. The server running this program contains a file named `"/sh"` that contains the 8-byte `SHELLCODE`, and no other files.

Each subpart lists a possible input to `fread` and the memory safety defenses that are enabled. Will the given input cause the program to execute shellcode?

Q3.1 (2 points) Defenses enabled: None

Input: `SHELLCODE`

- ○ Yes, immediately after calling `exec`
- ○ Yes, immediately after `vulnerable` returns
- ○ No, the program crashes immediately after calling `exec`
- ○ No, the program crashes immediately after `vulnerable` returns
- ○ No, the program returns from `vulnerable` without crashing or executing shellcode

Q3.2 (2 points) Defenses enabled: None

Input: `"/sh" + "\x00"`

- ○ Yes, immediately after calling `exec`
- ○ Yes, immediately after `vulnerable` returns
- ○ No, the program crashes immediately after calling `exec`
- ○ No, the program crashes immediately after `vulnerable` returns
- ○ No, the program returns from `vulnerable` without crashing or executing shellcode

Q3.3 (2 points) Defenses enabled: Non-executable pages only. Assume the program crashes the moment it tries to run any non-executable code.

Input: Same as the previous part. `"/sh" + "\x00"`

- ◯ Yes, immediately after calling `exec`
- ◯ Yes, immediately after `vulnerable` returns
- ◯ No, the program crashes immediately after calling `exec`
- ◯ No, the program crashes immediately after `vulnerable` returns
- ◯ No, the program returns from `vulnerable` without crashing or executing shellcode

Q3.4 (2 points) Defenses enabled: None

Input: `"STOP" + "\x00" + 11*"A" + "\x24\x10\xff\xff"`

- ◯ Yes, immediately after calling `exec`
- ◯ Yes, immediately after `vulnerable` returns
- ◯ No, the program crashes immediately after calling `exec`
- ◯ No, the program crashes immediately after `vulnerable` returns
- ◯ No, the program returns from `vulnerable` without crashing or executing shellcode

Q3.5 (2 points) Defenses enabled: None

Input: `"STOP" + "\x00"*4 + SHELLCODE + "A"*4 + "\x2c\x10\xff\xff"`

- ◯ Yes, immediately after calling `exec`
- ◯ Yes, immediately after `vulnerable` returns
- ◯ No, the program crashes immediately after calling `exec`
- ◯ No, the program crashes immediately after `vulnerable` returns
- ◯ No, the program returns from `vulnerable` without crashing or executing shellcode

Q3.6 (2 points) Defenses enabled: Non-executable pages only. Assume the program crashes the moment it tries to run any non-executable code.

Input: Same as the previous part.

`"STOP" + "\x00"*4 + SHELLCODE + "A"*4 + "\x2c\x10\xff\xff"`

- ◯ Yes, immediately after calling `exec`
- ◯ Yes, immediately after `vulnerable` returns
- ◯ No, the program crashes immediately after calling `exec`
- ◯ No, the program crashes immediately after `vulnerable` returns
- ◯ No, the program returns from `vulnerable` without crashing or executing shellcode

Q4 *Memory Safety: Ins and Outs* **(14 points)**

We'd like to write an exploit to execute shellcode with very high probability in the following code:

```
1  void inner() {
2      char* ptr;
3      char buf[12];
4      ptr = &buf;
5
6      fread(buf, 1, 12, stdin);
7      printf("%s", buf);
8      fread(buf, 1, 12, stdin);
9
10     // program pauses here for user to fill in the blank on Line 11
11     printf("%____u%hn");
12 }
13
14 void main() {
15     inner();
16 }
```

Assumptions:

- ASLR and stack canaries are enabled. All other defenses are disabled.
- Shellcode is located at address `0xdeadbeef`, and ASLR does not change the address of shellcode.

Post-exam note: Canaries are NOT enabled for this question – please ignore Q4.2.

Q4.1 (1 point) How many bytes are between the first byte of `buf` and the least-significant byte of the SFP of `inner`?

○ 12 ○ 15 ○ 20 ○ 23 ○ 28 ○ 31

Q4.2 (1 point) (DROPPED) Will leaking the stack canary be helpful in this exploit?

○ Yes, because `fread` can overwrite the canary with itself.

○ Yes, because `printf` can overwrite the canary with itself.

○ No, we need to use `fread` to write around the canary.

○ No, we need to use `printf` to write around the canary.

Q4.3 (2 points) What should we input to the `fread` call on Line 6?

- ○ `'A' * 12`, to remove all null bytes in `buf`.
- ○ `'A' * 12`, to cause a null byte to be written after `buf`.
- ○ `'\x00 * 12`, to remove all non-zero bytes in `buf` and `ptr`.
- ○ `'\xef\xad\xbe\xad' * 3`, to cause the `printf` call on Line 7 to leak the shellcode address.

Q4.4 (1 point) Next, the `printf` call on Line 7 executes. The resulting output string contains a numerical value that we'll need later. To get this value, we'll slice some bytes from the output and convert them to an integer, which we'll label `n`.

How should we slice the output to retrieve `n`?

○ `[0:4]` ○ `[4:8]` ○ `[8:12]` ○ `[12:16]` ○ `[16:20]`

For the rest of the question: If needed, you can perform arithmetic on the integer `n` (for example, you could write `n+7`), and you can assume that the resulting number is converted back to little-endian raw bytes for you.

Q4.5 (6 points) What should we input to the `fread` call on Line 8?

Your answer should be three 4-byte values, concatenated together. If a part of the input can be any 4-byte value, use `'A'*4` as the garbage bytes.

[ ] + [ ] + [ ]

Q4.6 (3 points) After Line 8, suppose the program pauses, and you get to fill in the blank on Line 11 while the program is paused. How do you compute the number to put in the blank?

First, compute:

[ ]

Then, treat the resulting number as little-endian raw bytes, and slice:

○ `[0:2]` ○ `[2:4]` ○ `[0:4]` ○ `[0:8]`

Q5 *Symmetric Cryptography: Meet Me in the Middle* **(11 points)**

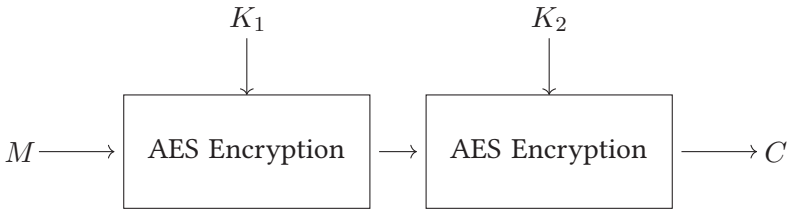
The Mallory Security Agency (MSA) is trying to spy on Alice and decrypt her messages.

Alice randomly selects K_1 and K_2 from a pool of 100 possible values.

Clarification during exam: all keys are chosen from separate pools. (However, this doesn't affect solutions, which consider worst-case scenarios)

Then, Alice encrypts an 128-bit message M with Double-AES, defined as

$$C = \mathsf{DAES}(K_1, K_2, M) = E_{K_2}(E_{K_1}(M))$$



Q5.1 (1 point) What is the decryption formula for Double-AES?

- ○ $D_{K_1}(D_{K_2}(C))$
- ○ $D_{K_2}(D_{K_1}(C))$
- ○ $E_{K_1}(E_{K_2}(C))$
- ○ $E_{K_2}(E_{K_1}(C))$

For the rest of the question, assume that the MSA knows the 100 possible values for K_1 and the 100 possible values for K_2.

Q5.2 (1 point) The MSA wants to naively brute-force all K_1, K_2 pairs. How many AES encryption/decryption calls are required to try all K_1, K_2 pairs?

- ○ 1
- ○ 200
- ○ 20000
- ○ 40000

The MSA plans to use a **meet-in-the-middle attack** to learn K_1 and K_2. We will design this attack in the next few subparts.

For this attack, assume the attacker has access to a known-plaintext pair (M, C).

1. Initialize a map A.
2. For each possible value of K_1, add this name-value pair to A:
   Name: [ANSWER TO Q5.3] Value: K_1
3. For each possible value of K_2, check if A contains the name [ANSWER TO Q5.4].
   If yes, output K_1 (from the name-value pair) and K_2.

Q5.3 (1 point) Answer to the first blank:

- ○ $E_{K_1}(M)$
- ○ $E_{K_1}(C)$
- ○ $D_{K_2}(M)$
- ○ $D_{K_2}(C)$
- ○ $E_{K_1}(E_{K_2}(M))$
- ○ $D_{K_1}(D_{K_2}(C))$

Q5.4 (1 point) Answer to the second blank:

- ○ $E_{K_1}(M)$
- ○ $D_{K_2}(M)$
- ○ $E_{K_1}(E_{K_2}(M))$
- ○ $E_{K_1}(C)$
- ○ $D_{K_2}(C)$
- ○ $D_{K_1}(D_{K_2}(C))$

Q5.5 (1 point) In the worst case, how many AES encryption/decryption calls are required in the attack above?

- ○ 1
- ○ 200
- ○ 400
- ○ 20000

Now consider a triple-AES scheme:

$$C = \mathsf{TAES}(K_1, K_2, K_3, M) = E_{K_3}(E_{K_2}(E_{K_1}(M)))$$

As before, assume that each key is randomly selected from a pool of 100 possible values.

Q5.6 (6 points) Provide an attack to recover K_1, K_2, K_3.

- Solutions using 100^3 or more AES encryption/decryption calls will receive 0 points.
- Solutions using 20100 calls in the worst case will receive up to 5/6 points.
- Solutions using 10200 calls in the worst case will receive up to full credit.

If you wish to leave this question blank and receive 0.5 points, fill in this bubble.

○ Please ignore what I write in the box below, and give me 0.5 points.

Q6 *Hash Functions: YAAS (Yet Another Authentication Scheme)* **(8 points)**

EvanBot decides to design a new authentication scheme.

Define `pwd` to be a secure password that only EvanBot knows.

Also, define H^k to be the result of repeatedly applying H, a cryptographically secure hash function, k times. Note: $H^0(x) = x$.

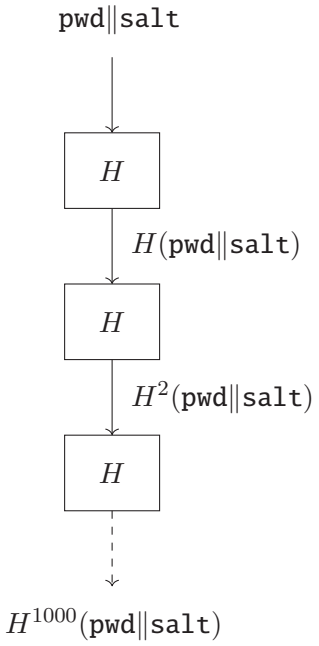
$$H^k(x) = \underbrace{H(H(...H}_{k \text{ times}}(x)))$$

To sign up:

1. EvanBot securely generates a 128-bit salt `salt`.
2. EvanBot sends $H^{1000}(\texttt{pwd}\|\texttt{salt})$ to the server.
3. The server maps EvanBot's username to a variable called `stored`. The server sets `stored` to be the value received in Step 2.

To log in for the n-th time (n starts at 1):

1. EvanBot sends $H^{1000-n}(\texttt{pwd}\|\texttt{salt})$ to the server.
2. The server checks whether [ANSWER TO Q6.1].
3. If Step 2 succeeds, the server updates `stored` to [ANSWER TO Q6.2].



Q6.1 (1 point) Let `Step1` be the value received in Step 1 of the login process. Select the correct option for the blank in Step 2.

- ○ $H(\texttt{Step1}) = \texttt{stored}$
- ○ $H(\texttt{stored}\|\texttt{salt}) = \texttt{Step1}$
- ○ $H(\texttt{stored}) = \texttt{Step1}$
- ○ $H(\texttt{Step1}\|\texttt{salt}) = \texttt{stored}$

Q6.2 (1 point) Select the correct option for the blank in Step 3.

- ○ `stored` (no update needed)
- ○ `Step1`
- ○ $H(\texttt{stored}\|\texttt{salt})$
- ○ $H^2(\texttt{Step1})$

Q6.3 (1 point) Does the server need to know `salt` in order to complete the login process?

- ○ Yes
- ○ No

Q6.4 (1 point) Eventually, EvanBot logs in by sending $H^{700}(\texttt{pwd}\|\texttt{salt})$. Given only `pwd`, `salt`, and $H^{700}(\texttt{pwd}\|\texttt{salt})$, how many calls to H does EvanBot need to make on the next login request?

- ○ 0
- ○ 1
- ○ 699
- ○ 700

Q6.5 (2 points) Eve is an on-path attacker.

Which of these sets of values, if seen by Eve, would allow Eve to learn the password? Each answer choice is independent.

- ☐ The first login attempt only
- ☐ The 1000th login attempt only
- ☐ Any two login attempts in a row
- ☐ All of the first 999 login attempts
- ☐ The 999th login attempt only
- ☐ All of the first 1000 login attempts
- ☐ None of the above

Q6.6 (2 points) Assume an attacker has compromised the server and can modify `stored`. Can the attacker login as EvanBot?

- ○ Yes, without knowing n, `pwd`, or `salt`.
- ○ Yes, but only if they know `salt`.
- ○ Yes, but only if they know n and `salt`.
- ○ Yes, but only if they know n.
- ○ No, even if they know n and `salt`.

Q7 *Web: Unscramble* (14 points)

`www.evanbook.com` is a website where users can submit and view posts. EvanBot is a user of this website, who is initially not logged in. Mallory is an on-path attacker between EvanBot and this website, and Mallory controls `www.mallory.com`.

- A user can load `www.evanbook.com/home` to see posts made by all users. (This behavior is the same whether the user is logged in or logged out.)
- A user can log in by making a POST request to `www.evanbook.com/login`, with their username and password (e.g. `"alice,password123"`) in the contents. If the username and password are correct, the HTTP response contains a session token cookie.
- A user who is logged in can load `www.evanbook.com/home?msg=X` to display all the posts, along with an additional message `X` at the top of the page.
- A user who is logged in can follow another user by making a GET request to `www.evanbook.com/follow?user=X`, replacing `X` with the username to follow.

In each subpart, provide a sequence of events (choosing from the list below) to execute the given attack. If you choose an event with a placeholder `X`, write the value you would insert into the placeholder.

A. EvanBot loads `www.evanbook.com/home`.
B. EvanBot loads `www.evanbook.com/home?msg=X`.
C. EvanBot makes a POST request with the correct username and password.
D. Mallory makes a post with contents `X`.
E. Mallory makes `www.mallory.com` send back `X`.
F. Mallory reads the HTTP request sent from EvanBot to `www.evanbook.com`.
G. Mallory reads the HTTP response sent from `www.evanbook.com` to EvanBot.

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 G) of the event. In the right box, if the event has a placeholder `X`, write the value you would use in the placeholder. If the event does not have a placeholder, leave the right box blank.

Example attack: Make EvanBot see the post "Mallory says hi."

Example answer: Mallory makes a post with contents "Mallory says hi." Then, EvanBot loads `www.evanbook.com/home`.

D	Mallory says hi
A	

Q7.1 (2 points) For this subpart, assume all requests are sent over HTTP (not HTTPS), and the session token cookie has attributes `Secure=false` and `HttpOnly=true`.

Attack: Learn the value of EvanBot's session token.

Q7.2 (2 points) Attack: Using stored XSS, make EvanBot run the JavaScript `alert(1)` with the origin of `www.evanbook.com`.

Q7.3 (2 points) From this subpart onwards, you may use the `post(url)` JavaScript function to send POST requests.

Attack: Make EvanBot log in as user `mallory` (who has password `161`).

Q7.4 (2 points) For this subpart, assume all requests are sent over HTTPS, and the session token cookie has attributes `Secure=true` and `HttpOnly=false`.

Attack: Use reflected XSS to learn the value of EvanBot's session token.

Q7.5 (3 points) Attack: Make EvanBot follow Mallory.

Q7.6 (3 points) Attack: Using stored XSS, make EvanBot run the JavaScript `alert(1)` with the origin of `www.mallory.com`.

Q8 *SQL Injection: Word Game* **(10 points)**

You're playing a word guessing game. Every day is numbered (e.g. today could be Day 75). The server has a unique secret word per day, and your goal is to guess the word.

The server contains a SQL table `answers` containing every day's secret word. The table has two columns: `day` (integer), and `word` (string).

When you enter a word, the server runs the following query, with `$input` replaced with the string you entered:

```
SELECT day FROM answers WHERE WORD = "$input"
```

If the query returns a single number equal to today's day number, then the server returns a webpage with a green checkmark. In all other cases, the server returns a webpage with a red X.

Q8.1 (2 points) For this subpart only, suppose today is Day 75, and tomorrow is Day 76. Select all inputs that would check whether tomorrow's word is `pancake`.

HINT: All options have valid SQL syntax.

- ☐ `" UNION SELECT 76 FROM answers WHERE word = "pancake" AND day = 76--`
- ☐ `" UNION SELECT day FROM answers WHERE word = "pancake`
- ☐ `" UNION SELECT day+1 FROM answers WHERE word = "pancake`
- ☐ `" UNION SELECT "pancake" FROM answers WHERE day = 76--`
- ☐ None of the above

Q8.2 (2 points) For this subpart only, you don't know today's day number, but you know that tomorrow's day number is today's day number plus one. Without using semicolons, provide an input that can check whether tomorrow's word is `pancake`.

For the rest of the question, consider an updated version of the word game.

Every word is 5 characters, and players know this.

1. Each day, a player inputs 5 strings, one for each character of their single guess. Each string should contain only one character, but malicious users might input longer strings.
2. For input `i` (where $1 \leq$ `i` ≤ 5), the server performs the following two steps, with `$input` replaced with the user input, and `i` replaced by the current input number.
3. First, the server checks whether the `i`th character of the word is equal to the user's guess:

   ```
   SELECT day FROM answers WHERE SUBSTRING(word, i, 1) = "$input"
   ```

   If the values returned by this first query include today's day number, then the server displays a green box in position `i`.
4. Otherwise, the server then performs a second check to see whether the user's guess exists anywhere in today's word:

   ```
   SELECT day FROM answers WHERE CHARINDEX("$input", word) > 0
   ```

   If the values returned by this second query include today's day number, then the server displays a yellow box at position `i`.

Q8.3 (3 points) Without using semicolons or the `SELECT` keyword, provide a value for the `i`th input that would cause the `i`th position to display green when the letter `h` appears anywhere in today's word.

Q8.4 (2 points) Without using semicolons or the `SELECT` keyword, provide a value for the `i`th input that would cause the `i`th position to display yellow when today's word is `bacon`.

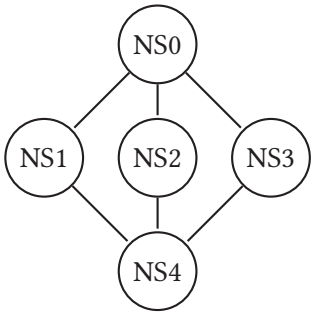
Q8.5 (1 point) For this subpart only, the server implements a check to verify that each of the 5 input strings is only one character long. Will this stop all SQL injection attacks?

- ○ Yes, because all injections must end in `--`, which is two characters.
- ○ Yes, because one character is not enough to add extra logic to the query.
- ○ No, because a 1-character input exists that would cause the query to crash.
- ○ No, because the injection could be split across the 5 inputs.

Q9 *DNS: Double-Check Your Work* **(7 points)**

The IP address of `eecs.berkeley.edu` is `5.5.5.5`. EvanBot does not know this, but would like to use DNSSEC to learn this IP address.

Consider the following DNS name server hierarchy.



	Zone	Domain	IP
NS0	`.` (root)	`ns0.net`	`0.0.0.0`
NS1	`.edu`	`ns1.net`	`1.1.1.1`
NS2	`.edu`	`ns2.net`	`2.2.2.2`
NS3	`.edu`	`ns3.net`	`3.3.3.3`
NS4	`berkeley.edu`	`ns4.net`	`4.4.4.4`

Q9.1 (2 points) In real life, the 3 `.edu` name servers would all use the same public/private key pair.

Name one reason why having multiple name servers for a zone is useful, even if they all use the same key pair. You can answer in 10 words or fewer. The staff answer is one word.

For the rest of the question, assume that every name server has its own unique public/private key pair.

EvanBot wants multiple verifications of the IP address of `eecs.berkeley.edu`.

First, EvanBot queries the root name server for information about all 3 `.edu` name servers.

Q9.2 (1 point) How many A type records are returned by the root name server?

○ 0 ○ 1 ○ 3 ○ 4 or more

Next, EvanBot queries all 3 `.edu` name servers for information about the `berkeley.edu` name server.

Q9.3 (2 points) Select all IP addresses that appear in the records returned by the 3 `.edu` name servers.

☐ `0.0.0.0` ☐ `4.4.4.4` ☐ None of the above
☐ `1.1.1.1` ☐ `5.5.5.5`

Q9.4 (1 point) How many different DNSKEY records are returned by the 3 `.edu` name servers?

○ 0 ○ 1 ○ 3 ○ 4 or more

Q9.5 (1 point) Finally, EvanBot queries the `berkeley.edu` name server. In total, how many different RRSIG records has EvanBot received from all the name servers during this DNS query?

Note: Two RRSIG records are different if the digital signatures in the records are not equal.

○ 0 ○ 1 ○ 3 ○ 4 or more

Q10 *Networking: A TORrible Mistake* **(7 points)**

Q10.1 (1 point) Assuming no malicious nodes collude, an n-node Tor circuit provides anonymity (i.e. no node learns who both the user and server are) when at least _____ node(s) are honest. Fill in the blank.

○ 0 ○ 1 ○ $n-1$ ○ n

For the next 3 subparts, a user is using Tor to send a message to a server. Assume that there is no collusion between any Tor nodes, and that the user choses exactly 3 nodes for their Tor circuit.

Q10.2 (1 point) Which values can a malicious **entry** node learn? Select all that apply.

☐ The IP address of the user ☐ The list of all nodes in the circuit
☐ The IP address of the server ☐ None of the above

Q10.3 (1 point) Which values can a malicious **exit** node learn? Select all that apply.

☐ The IP address of the user ☐ The list of all nodes in the circuit
☐ The IP address of the server ☐ None of the above

Q10.4 (1 point) Which values can an on-path attacker on the user's local network learn? Select all that apply.

☐ The IP address of the user ☐ The list of all nodes in the circuit
☐ The IP address of the server ☐ None of the above

When a new user first downloads Tor, they need to download a list of nodes from a trusted directory server.

A malicious, on-path attacker on the user's local network wishes to eavesdrop on the new user's Tor connection. Assume that the attacker controls 3 nodes out of 100 total Tor nodes, and can win any data race.

For the next three subparts, select the approximate probability that the attacker can learn the identity of the server.

Q10.5 (1 point) User connects to the directory via TLS, attacker is on-path.

○ Exactly 0% ○ Greater than 50%, less than 100%
○ Greater than 0%, less than 50% ○ Exactly 100%

Q10.6 (1 point) User connects to the directory via TCP, attacker is on-path.

○ Exactly 0% ○ Greater than 50%, less than 100%
○ Greater than 0%, less than 50% ○ Exactly 100%

Q10.7 (1 point) User connects to the directory via TCP, attacker is off-path.

○ Exactly 0% ○ Greater than 50%, less than 100%
○ Greater than 0%, less than 50% ○ Exactly 100%

Q11 *Networking: New Phone Who This* **(9 points)**

EvanBot joins a new **broadcast** local network with many users. CodaBot is on the local network, but EvanBot doesn't know CodaBot's phone number. EvanBot wants to learn CodaBot's phone number, using the following protocol:

1. EvanBot broadcasts a request asking what CodaBot's phone number is.
2. CodaBot sends a response to EvanBot with their phone number.
3. EvanBot caches the phone number.

Q11.1 (1 point) Which networking protocol is this most similar to?

○ ARP ○ WPA2 ○ BGP ○ TCP

Q11.2 (2 points) Eve is an on-path attacker in the local network. Select all attacks that Eve can carry out.

☐ Perform an online brute-force attack to learn CodaBot's phone number, by sending back every possible phone number to EvanBot.

☐ Learn CodaBot's phone number by reading message(s) Eve was not supposed to read.

☐ Learn CodaBot's phone number without reading message(s) Eve was not supposed to read.

☐ Convince EvanBot that CodaBot's phone number is some malicious value chosen by Eve.

☐ None of the above

In the next three subparts, consider this modification to the protocol: Instead of sending just the phone number, CodaBot sends their public key, and a signature on their phone number.

When EvanBot receives this data, EvanBot uses the public key to verify the signature on the phone number.

Eve wants to trick EvanBot into thinking CodaBot's phone number is a malicious value chosen by Eve. What values does Eve include in the packet she sends to EvanBot?

Q11.3 (1 point) For the public key, Eve sends:

○ Eve's public key ○ EvanBot's public key

○ CodaBot's public key ○ The router's public key

Q11.4 (1 point) For the signature over the phone number, Eve signs using:

○ Eve's private key ○ EvanBot's private key

○ CodaBot's private key ○ The router's private key

Q11.5 (1 point) How often will this attack succeed?

○ 100% of the time ○ Only when CodaBot's packet arrives first

○ Only when Eve's packet arrives first ○ Never

For the rest of the question, consider a different modification: we send all messages over TLS instead.

Q11.6 (1 point) How should Step 1 be modified?

- ○ Form one TLS connection, and broadcast the request.
- ○ Form one TLS connection with each person on the local network, and then send the request directly to each person.
- ○ Form one TLS connection with the router. Then, broadcast the request, encrypted and MACed with the symmetric keys from the connection with the router.

Q11.7 (2 points) EvanBot wants to think about some possible disadvantages of this modification. Select all true statements.

- ☐ Adding TLS makes this protocol slower.
- ☐ Adding TLS requires each user on the network to have a certificate for themselves.
- ☐ Eve can learn EvanBot's identity.
- ☐ Eve can learn CodaBot's phone number.
- ☐ None of the above

Everything below this line will not be graded.

Post-Exam Activity: Vacation

Where are the 161 bots traveling to this winter break?

Comment Box

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