CS 161 Spring 2024 — Introduction to Computer Security — Final

Name: _____

Student ID: _____

This exam is 170 minutes long.

Question:	1	2	3	4	5	6	7	8	9	Total
Points:	0	10	13	15	13	11	17	11	10	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.

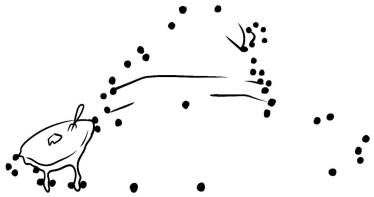
- ■ You can select
- ■ multiple squares (completely filled)

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:
(Just for fun, not graded.)

Connect the dots to form a drawing!
(I wonder what the result will be...)

Bonus: Once you're done, draw something below it!



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* **(10 points)**

Each true/false is worth 0.5 points.

Q2.1 The Caltopian Army designs a secure radio that encodes cryptographic keys into physical keys. To access a specific secure channel, the user inserts a key with a corresponding color.

True or False: This is an example of Consider Human Factors.

○ True ○ False

Q2.2 EvanBook Inc. requires the approval of two separate engineers to unlock the server room.

True or False: This is an example of Shannon's Maxim.

○ True ○ False

Q2.3 True or False: Stack canaries are the same across different functions within the same program execution.

○ True ○ False

Q2.4 True or False: AES-CBC is often said to act like a stream cipher.

○ True ○ False

Q2.5 True or False: An attacker expects to have to try 2^{196} different values to find a collision in a 256-bit secure cryptographic hash function.

○ True ○ False

Q2.6 True or False: RSA keys are usually chosen to be either 128 or 256 bits long.

○ True ○ False

Q2.7 True or False: It is possible to have multiple valid certificates for a single website.

○ True ○ False

Q2.8 True or False: `www.google.com` and `google.com` have different origins under the Same-Origin Policy.

○ True ○ False

Q2.9 True or False: CSRF tokens are often stored in cookies.

○ True ○ False

Q2.10 True or False: It is possible for a cookie to have both `HttpOnly` and `Secure` set to true.

○ True ○ False

Q2.11 True or False: Stored XSS attacks are often more severe than reflected XSS attacks, because users do not need to click on an attacker-controlled link in stored XSS attacks.

○ True ○ False

Q2.12 True or False: A MITM attacker can force the user's browser to execute malicious JavaScript by tampering with the HTTP response.

○ True ○ False

Q2.13 True or False: ARP spoofing requires an off-path attacker to correctly guess the source port of the ARP request sender.

○ True ○ False

Q2.14 True or False: The correct order of the DHCP handshake is: Client Discover, DHCP Acknowledgement, Client Request, DHCP Offer.

○ True ○ False

Q2.15 An attacker who manages to impersonate a WPA2 access point has full access to the contents of the HTTPS requests from the network clients.

○ True ○ False

Q2.16 Every BGP AS requires a certificate signed by a certificate authority.

○ True ○ False

For the next two subparts, you are a consultant for a large corporation that wishes to install intrusion detection equipment.

Q2.17 True or False: It would likely be cheaper to install a NIDS instead of installing HIDS.

○ True ○ False

Q2.18 True or False: Installing a NIDS would allow the corporation to easily inspect the contents of the employees' HTTPS requests.

○ True ○ False

Q2.19 True or False: Double spending on the Bitcoin network is only possible by stealing the private key of another user.

○ True ○ False

Q2.20 True or False: LLMs are vulnerable to prompt injection attacks because they generally cannot distinguish between input and commands.

○ True ○ False

Q2.21 (0 points) True or False: `EvanBot is a real bot.`

○ True ○ False

Q3 *Memory Safety: Everyone Loves PIE* **(13 points)**

Consider the following vulnerable C code:

```
1  void cake() {
2    char buf[8];
3    char input[9];
4    int i;
5
6    fread(input, 9, 1, stdin);
7
8    for (i = 8; i >= 0; i--) {
9      buf[i] = input[i];
10   }
11   return;
12 }
13
14 void pie() {
15   char cookies[64];
16
17   // Prints out the 4-byte address of cookies
18   printf("%p", &cookies);
19
20   fgets(cookies, 64, stdin);
21   cake();
22   return;
23 }
```

Stack at Line 6

RIP of `pie`
SFP of `pie`
(1)
RIP of `cake`
(2)
`buf`
(3)
`i`

Assumptions:

- `SHELLCODE` is 63 bytes long.
- ASLR is enabled. All other defenses are disabled.

Q3.1 (1 point) What values go in blanks (1) through (3) in the stack diagram above?

- ○ (1) `&p` (2) SFP of `cake` (3) `SFP of printf`
- ○ (1) `cookies` (2) SFP of `cake` (3) `input`
- ○ (1) `cookies` (2) SFP of `cake` (3) `RIP of fgets`
- ○ (1) `RIP of printf` (2) SFP of `printf` (3) input

Q3.2 (1 point) Which vulnerability is present in the code?

- ○ Off-by-one
- ○ Signed/unsigned vulnerability
- ○ Format string vulnerability
- ○ Time-of-check to time-of-use

In the next two subparts, you will provide inputs to cause `SHELLCODE` to execute with high probability.

Let `OUT` be the output from the `printf` call on Line 18. Assume that you can slice this value (e.g. `OUT[0:2]` returns the 2 least significant bytes of `&cookies`). You may also perform arithmetic on this value (e.g. `OUT[0:2] + 4`) and assume it will be converted to/from the correct types automatically.

Q3.3 (2 points) Provide a value for the `fgets` call on Line 20.

Q3.4 (5 points) Fill in each blank with an integer to provide an input to the `fread` call on Line 6.

You must put an integer for every blank even if the final slice would be equivalent – for example, you must put both "0" and "7" in the blanks for `OUT[0:7]`, even though `OUT[:7]` is equivalent.

Note that the `+` between terms refers to string concatenation (like in Project 1 syntax), but the minus sign in the third term refers to subtracting from the `OUT[_:_]` value.

```
'A'*________ + OUT[______:______] + (OUT[______:______] - ________)
```

Q3.5 (2 points) Which of these defenses, if enabled by itself, would prevent the exploit (without modifications) from working? For pointer authentication only, assume the program runs on a 64-bit system.

- ☐ Stack canaries
- ☐ Pointer authentication
- ☐ Non-executable pages
- ☐ None of the above

Q3.6 (2 points) Which of these variable values would cause the exploit to break?

- ○ RIP of `pie` = `0x10c3fa00`
- ○ RIP of `cake` = `0x10237acf`
- ○ address of `cookies` = `0xffff5fc0`
- ○ SFP of `cake` = `0xffffcd04`

Q4 *Memory Safety: Breaking Bot* **(15 points)**

EvanBot has decided to manufacture an industrial amount of pancakes and has gone to Walter White for help. Consider the following vulnerable C code:

```
1  typedef struct {
2    char name[12];
3    void (*task)(); // task is a pointer to a function
4  } person;
5
6  /* implementations not shown */
7  void cook() { ... };
8  void sell() { ... };
9
10 void rv() {
11     person *p;
12     char *formula;
13     char saved_name[12];
14
15     p = (person *) malloc(sizeof(person));
16     fgets(p->name, 12, stdin);
17     if (strcmp(p->name, "Walter") == 0) {
18         p->task = cook;
19     } else {
20         p->task = sell;
21     }
22     strlcpy(saved_name, p->name, 12);
23     free(p);
24
25     formula = (char *) malloc(17);
26     fgets(formula, 17, stdin);
27
28     p->task();
29 }
```

Stack at Line 15

RIP of `rv`
(1)
(2)
(3)
`saved_name`

Assumptions:

- The heap starts at address `0x30000000` and grows upwards.
- `malloc` always allocates starting at the lowest possible address with enough free space.
- `malloc` always allocates the exact amount of memory required by its input, with no metadata.
- Your goal is to call `system("/bin/sh")`, which will spawn a shell.
- The function `system` is located in memory at address `0x08120161`.
- The address of `saved_name` is `0xffffca10`.
- Non-executable pages are enabled. All other defenses are disabled.

EvanBot says you should go re-read the assumptions before proceeding!

Q4.1 (1 point) What values go in blanks (1) through (3) in the stack diagram above?

- ○ (1) SFP of `rv` (2) `p` (3) `formula`
- ○ (1) SFP of `rv` (2) `formula` (3) `p`
- ○ (1) `formula` (2) `p` (3) RIP of `cook`
- ○ (1) `formula` (2) RIP of `cook` (3) SFP of `cook`

Q4.2 (1 point) Which vulnerability is present in the code?

- ○ Off-by-one
- ○ Format string vulnerability
- ○ Signed/unsigned vulnerability
- ○ Use after free

Q4.3 (1 point) What address is stored in the variable `formula` after Line 25 is executed?

- ○ `0xffffca10`
- ○ `0xffffca1c`
- ○ `0x30000000`
- ○ `0x30000010`

In the next two subparts, provide inputs that would cause the program to execute `system("/bin/sh")`.

If a part of the input can be any non-zero value, use `'A'*n` to represent the `n` bytes of garbage.

Q4.4 (4 points) Input to `fgets` at Line 16:

Q4.5 (6 points) Input to `fgets` at Line 26:

Q4.6 (1 point) Would it still be possible for your exploit to work (without modifications) if stack canaries are enabled?

- ○ Yes, because the exploit writes around the canary to overwrite values above the canary.
- ○ Yes, because the exploit never tries overwriting values above the canary.
- ○ No, because we cannot leak the canary value before overwriting it.
- ○ No, because the least-significant byte of the canary is overwritten by a null byte.

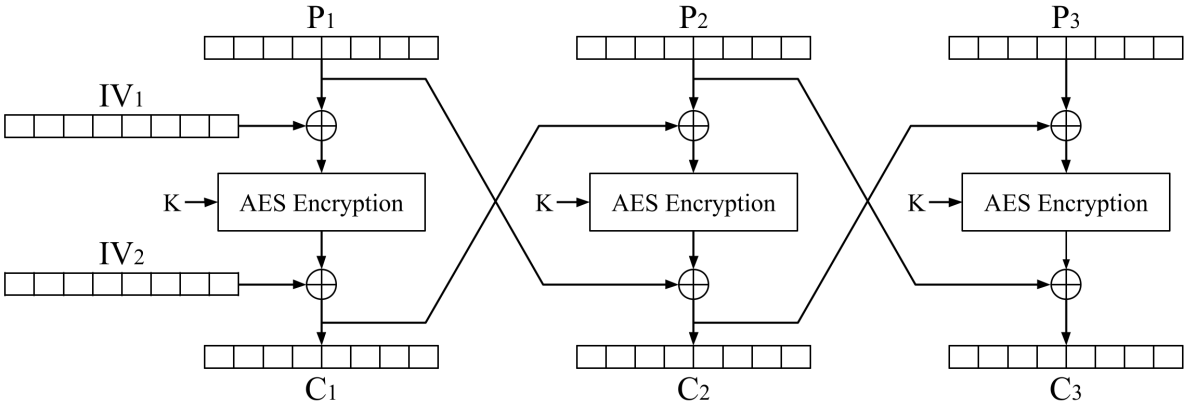
Q4.7 (1 point) Would it still be possible for your exploit to work with high probability (without modifications) if ASLR is enabled, assuming the code section is randomized?

- ○ Yes
- ○ No

Q5 *Symmetric Cryptography: AES-ROVW* **(13 points)**

EvanBot designs the AES-ROVW mode of operation as follows:

$$C_1 = E_K(P_1 \oplus IV_1) \oplus IV_2$$
$$C_i = E_K(P_i \oplus C_{i-1}) \oplus P_{i-1} \text{ (for } i \geq 2)$$



Q5.1 (1 point) Select the decryption formula for P_i, for $i \geq 2$.

- ○ $P_i = D_K(C_{i-1} \oplus P_i) \oplus C_i$
- ○ $P_i = D_K(C_i) \oplus C_{i-1} \oplus P_{i-1}$
- ○ $P_i = D_K(C_i \oplus P_{i-1}) \oplus C_{i-1}$
- ○ $P_i = D_K(C_i \oplus C_{i-1}) \oplus P_{i-1}$

Q5.2 (1 point) Select all true statements.

- ☐ Encryption is parallelizable.
- ☐ Decryption is parallelizable.
- ☐ None of the above

Q5.3 (3 points) Select all true statements.

- ☐ AES-ROVW is IND-CPA secure if IV_1 and IV_2 are independently randomly generated.
- ☐ AES-ROVW is IND-CPA secure if IV_1 is randomly generated and $IV_2 = H(IV_1)$.
- ☐ AES-ROVW is IND-CPA secure if IV_2 is randomly generated and $IV_1 = H(IV_2)$.
- ☐ None of the above.

Alice has a two-block message (P_1, P_2). Alice encrypts this message with AES-ROVW to get (IV_1, IV_2, C_1, C_2).

Mallory, a MITM attacker, intercepts Alice's ciphertext (IV_1, IV_2, C_1, C_2), and Mallory **knows the original plaintext value** (P_1, P_2).

Mallory wants to change the ciphertext to $(IV_1', IV_2', C_1', C_2')$, such that when Bob receives the modified ciphertext and decrypts it, he sees (P_1', P_2'), a malicious message of Mallory's choosing.

In the next four subparts, give the values for Mallory's tampered ciphertext $(IV_1', IV_2', C_1', C_2')$. Select as many options as you need.

Q5.4 (2 points) IV_1' is equal to these values, XORed together.

For example, if you think $IV_1' = P_2 \oplus C_2$, then bubble in P_2 and C_2.

☐ IV_1 ☐ P_1 ☐ P_1' ☐ C_1

☐ IV_2 ☐ P_2 ☐ P_2' ☐ C_2

Q5.5 (2 points) IV_2' is equal to these values, XORed together.

☐ IV_1 ☐ P_1 ☐ P_1' ☐ C_1

☐ IV_2 ☐ P_2 ☐ P_2' ☐ C_2

Q5.6 (2 points) C_1' is equal to these values, XORed together.

☐ IV_1 ☐ P_1 ☐ P_1' ☐ C_1

☐ IV_2 ☐ P_2 ☐ P_2' ☐ C_2

Q5.7 (2 points) C_2' is equal to these values, XORed together.

☐ IV_1 ☐ P_1 ☐ P_1' ☐ C_1

☐ IV_2 ☐ P_2 ☐ P_2' ☐ C_2

Q6 *Asymmetric Cryptography: Plentiful Playlists* **(11 points)**

Alice and Bob wish to create a music playlist for their upcoming road trip to Pittsburgh. Alice and Bob each come up with a list of n songs. Some (but not all) of the songs might appear on both lists.

Alice and Bob want to learn the songs that are on both lists, without revealing their individual lists to each other.

For example, say Alice's list has the songs "One Little Victory" and "Motivation", while Bob's list has "Motivation" and "Give It All". Both Alice and Bob should be able to learn that both lists contain "Motivation". However, Alice should learn nothing about Bob's other songs, and vice versa.

They decide to use the following protocol, but need your help to fill in the blanks. Assume that p is a large prime, like those used in Diffie-Hellman, and each song is represented as an integer $\bmod p$.

1. Alice and Bob denote their lists as $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$ respectively.
2. Alice generates a random number $r \pmod p$.
3. Bob generates a random number $k \pmod p$.
4. For each element a_i in Alice's list, Alice sends $\texttt{STEP4}_i = a_i^r \pmod p$ to Bob.
5. For each element $\texttt{STEP4}_i$ received from Alice in the previous step, Bob computes $\texttt{STEP5}_i =$ ____, and sends $\texttt{STEP5}_i$ to Alice.
6. For each element $\texttt{STEP5}_i$ received from Bob in the previous step, Alice computes $\texttt{STEP6}_i =$ ____.
7. For each element b_i in Bob's list, Bob sends $\texttt{STEP7}_i =$ ____ to Alice.
8. Alice compares the set of $\texttt{STEP6}_i$ (for all i) and $\texttt{STEP7}_j$ (for all j) to find matching pairs, i.e. all (i, j) such that $\texttt{STEP6}_i = \texttt{STEP7}_j$. For the pairs that match, Alice finds the corresponding a_i.
9. Alice sends the set of all matching a_i to Bob over a secure channel.

Q6.1 (2 points) Replace the blank in Step 5.

- ○ $(\texttt{STEP4}_i) - k \pmod p$
- ○ $\texttt{STEP4}_i + k \pmod p$
- ○ $k \cdot \texttt{STEP4}_i \pmod p$
- ○ $(\texttt{STEP4}_i)^k \pmod p$.

Q6.2 (2 points) Define r^{-1} such that $g^{r \cdot r^{-1}} \equiv g \pmod p$ for all $g \pmod p$. Assume that such an r^{-1} always exists.

Replace the blank in Step 6.

- ○ $(\texttt{STEP5}_i)^{r^{-1}} \pmod p$
- ○ $(\texttt{STEP5}_i)^{r \cdot r^{-1}} \pmod p$
- ○ $r^{-1} \cdot (\texttt{STEP5}_i) \pmod p$
- ○ $(\texttt{STEP5}_i)^r \pmod p$

Q6.3 (2 points) Replace the blank in Step 7.

- ○ $b_i^r \pmod p$
- ○ $b_i + k \pmod p$
- ○ $b_i^k \pmod p$
- ○ $(b_i^r)^k \pmod p$

Q6.4 (1 point) Which option best explains why Alice and Bob cannot learn the songs in the other person's list (besides the songs that are in both lists)?

- ○ The values are encrypted with a shared symmetric key derived via Diffie-Hellman.
- ○ Alice does not know k (Bob's random number), and Bob does not know r (Alice's random number).
- ○ Exponentiation is commutative modulo p, i.e. $(x^a)^b \equiv x^{ab} \equiv (x^b)^a \pmod{p}$.
- ○ Factoring the product of two large primes is considered to be difficult.

Q6.5 (2 points) What information is leaked to a third-party eavesdropper? Select all that apply.

- ☐ All of Alice's songs
- ☐ All of Bob's songs
- ☐ All songs on both lists
- ☐ The number of songs in Alice's list
- ☐ The number of songs in Bob's list
- ☐ None of the above

Q6.6 (2 points) In the current scheme, a third-party eavesdropper learns some information about the number of songs on both lists.

Which option(s) prevent the eavesdropper from learning any information about the number of songs on both lists? Assume there are ℓ songs on both lists. Reminder: There are n songs on each list.

- ☐ Add $n - \ell$ dummy values to the message in Step 9.
- ☐ Add $\min(n, \ell)$ dummy values to the message in Step 9.
- ☐ Add $\max(n, \ell)$ dummy values to the message in Step 9.
- ☐ Add $n + \ell$ dummy values to the message in Step 9.
- ☐ None of the above

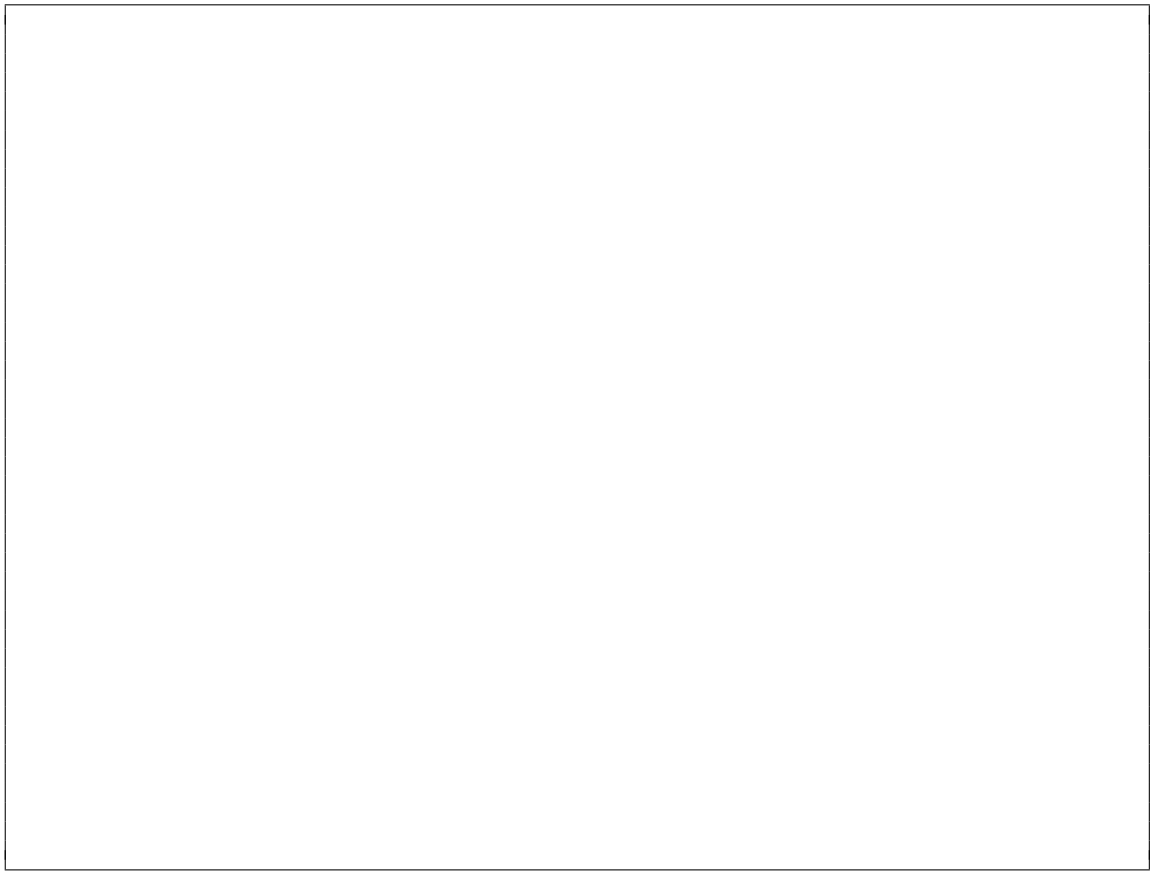
Q6.7 (0 points) **A+ Question: This subpart is worth no points and is considerably more difficult than the rest of the exam. If you correctly solve this subpart and are in the "A" grade bin at the end of the semester, you may be moved up to the "A+" bin.** .

In this scenario, you are controlling Alice who is running the aforementioned protocol with Bob. Design an attack to recover Bob's ephemeral key $k \pmod{p-1}$, and by extension reveal his entire set.

Assumptions:

1. Assume the correct answers were selected for the first three subparts (if you did not select the correct responses, you will receive no credit for this subpart).
2. You may choose an arbitrary set of elements for Alice, and also deviate from the protocol in any of Alice's steps – for example, you may set $r = 1$ in Step 2. Bob will follow the protocol honestly and will not try to actively detect any sort of attack.
3. $p = p_1 p_2 \cdots p_n + 1$, where each individual prime p_i is **small enough that you can solve the discrete logarithm problem** $\bmod p_i$. The attacker knows this factorization.
4. k is invertible modulo $p - 1$.

HINT: Use the Chinese Remainder Theorem.



Q7 *Web Security: Suspicious SQL* **(17 points)**

A bank website, `bank.com`, decides to test out a new form of user authentication.

- **When a user signs up**: The user chooses an alphanumeric username. The bank securely gives the user a symmetric key (known only to the user and the bank).

  The bank has a SQL table named `keys`, which maps usernames to keys. The table has two columns: `username` (of type string) and `key` (of type integer).
- **When a sender wants to send money**: the sender encrypts the recipient's username with the sender's key, and makes a GET request to:

  `www.bank.com/transfer?sender=_____&recipient=_____`

  The first blank contains the sender's username. The second blank contains the recipient's username, encrypted with the sender's key.
- **When the bank receives a GET request**: the bank first runs this SQL query:

  `SELECT key FROM keys WHERE username='_____'`

  where the blank is replaced with the first URL parameter.

  Then, the bank will use the key returned by the query to decrypt the the recipient's username.

  Finally, the bank transfers money from the sender (whose username comes from the first URL parameter) to the recipient (whose username comes from the second URL parameter, decrypted).

Note: For this entire question, you do not need to consider URL escaping.

Q7.1 (1 point) If requests to `bank.com` are made over HTTP, which of these values can an on-path attacker see? Select all that apply.

- ☐ Sender's IP address
- ☐ Recipient's plaintext username
- ☐ Sender's plaintext username
- ☐ None of the above

Q7.2 (1 point) If requests to `bank.com` are made over HTTPS, which of these values can an on-path attacker see? Select all that apply.

- ☐ Sender's IP address
- ☐ Recipient's plaintext username
- ☐ Sender's plaintext username
- ☐ None of the above

For the next four subparts: Mallory (username: `mallory`) wants to cause the user Bob (username: `bob`) to run the Javascript function `hack()`, once he clicks on a `bank.com` link provided by Mallory.

In each subpart, select whether Mallory's attack is possible. If you select "Yes," provide the URL parameters (`sender` and `recipient`) in the link that Bob clicks. In your answer(s), you may use Mallory's key K_{mallory} and the encryption function $\mathsf{Enc}(\cdot,\cdot)$, where the first argument is the key and the second argument is the plaintext.

Q7.3 (3 points) For this subpart only: If the decrypted recipient username does not exist in the `keys` table, the bank will return an HTML page with the text:

"____ does not exist"

replacing the blank with the decrypted recipient username.

Can Mallory cause Bob to call `hack()`?

○ Yes ○ No

If you selected "Yes":

`sender`:

`recipient`:

Q7.4 (3 points) For this subpart only: if the decrypted recipient username does not exist in the `keys` table, the bank will return an HTTP 404 response.

Can Mallory cause Bob to call `hack()`?

○ Yes ○ No

If you selected "Yes":

`sender`:

`recipient`:

Q7.5 (1 point) What type of attack is Mallory trying to execute in the previous two subparts?

- ○ Stored XSS
- ○ Reflected XSS
- ○ SQL injection
- ○ CSRF

Q7.6 (1 point) Select the true statement about Mallory's attack (assuming it succeeds).

Assume `mallory.com` is a website controlled by Mallory.

- ○ The `hack()` function runs with the origin of `bank.com`.
- ○ The `hack()` function runs with the origin of `mallory.com`.
- ○ `hack()` is able to read cookies with the HttpOnly flag set.
- ○ Instead of making Bob click on a `bank.com` link, Mallory could make Bob click on a link like `mallory.com/hack`, and the attack would be the same.

In the rest of the question, Mallory wants to create a list of one or more URLs such that when **Mallory** clicks on every URL in the list, one after the other, the bank sends money from Bob to Mallory.

Q7.7 (1 point) Mallory thinks that she could create a single URL to execute this attack.

Why is it not possible to execute this attack with a single URL?

- ○ The SQL injection has to be placed in `sender`, but this will cause the sender username to be invalid.
- ○ The SQL injection has to be placed in `recipient`, but this will cause the recipient username to be invalid.
- ○ The SQL injection has to be placed in both `sender` and `recipient`, but this causes both fields to be invalid.
- ○ The SQL injection needs to be split across two separate URLs.

Q7.8 (6 points) Construct two URLs, such that when Mallory clicks on the first URL, and then the second URL, the bank will send money from username `bob` to username `mallory`.

You may use K_{mallory} and the encryption function $\mathsf{Enc}(K, M)$ in your answer. If a value could be anything, you must write the word `"anything"` in the box.

First URL:

`sender`:

`recipient`:

Second URL:

`sender`:

`recipient`:

Q8 *DNS: Check Please* **(11 points)**

Suppose that in DNS, we introduce a single, additional name server that can be used to check if the records returned by other name servers are correct.

Every time the user makes a DNS query, the user also sends the same query to the check server. The user receives the answer directly from the check server, and can compare that answer against the answer received from the other name servers.

Q8.1 (1 point) The zone of the check server must be set to `.` (root). Otherwise, _____ would cause records from the check server to be rejected. What term goes in the blank?

- ○ Kaminsky attack
- ○ NSEC3 hashing
- ○ Bailiwick checking
- ○ Glue validation

Q8.2 (1 point) Is the Kaminsky attack still possible when users also issue a request to the check server for every query?

- ○ Yes, assuming the user doesn't validate glue records with the check server.
- ○ Yes, even if the user validates glue records with the check server.
- ○ No, because the check server is in the root zone.
- ○ No, because in the Kaminsky attack, the malicious record is returned in the Answer section.

For the rest of question, suppose we're now using DNSSEC instead of DNS.

Q8.3 (2 points) Which of these options, by itself, would ensure that every recursive resolver trusts the check server? Select all that apply.

- ☐ The root name server provides a DS and RRSIG record endorsing the check server.
- ☐ The check server sends an RRSIG record over its own public key.
- ☐ The check server sends an RRSIG record over every record it sends.
- ☐ The check server's public key is hard-coded in all resolvers.
- ☐ None of the above

For the rest of the question, suppose we're using DNSSEC, and every client trusts the check server.

Q8.4 (1 point) Which record type(s) does the check server need to send to securely answer the user's query? Select all that apply.

- ☐ A type record
- ☐ DS type record
- ☐ NS type record
- ☐ None of the above
- ☐ RRSIG type record

Q8.5 (2 points) For this subpart only, assume an attacker has compromised the check server.

Also, assume that the user will re-send the query if they receive different answers from the check server and the other name servers.

Select all attacks that this attacker could carry out.

- ☐ DoS attack
- ☐ None of the above
- ☐ Cache poisoning attack

Q8.6 (2 points) Select all true statements about the check server verifying non-existent domain names.

- ☐ If the server uses **offline** signing, it will need to store a large amount of NSEC records.
- ☐ If the server uses **online** signing, it will need to store a large amount of NSEC records.
- ☐ If the server uses **online** signing, it is more vulnerable to DoS attacks (compared to using offline signing).
- ☐ If the server uses **online** signing, it is more vulnerable to having its private key stolen (compared to using offline signing).
- ☐ None of the above

Q8.7 (2 points) Name one usability disadvantage to having a single additional check server. You can answer in 10 words or fewer (the staff answer is 1 word).

Q9 *TLS: Key Rotation* **(10 points)**

Consider modifying TLS so that within a long-running connection, the server and client switch to using a different set of symmetric keys every hour.

For the entire question, you can assume TLS does **not** use record numbers.

Q9.1 (2 points) Suppose the server and client switch to using a different randomly-generated key every hour. Select all true statements.

- ☐ Within one connection, a MITM attacker can only replay a message from the same hour.
- ☐ A MITM can replay messages from an earlier connection, if it was made during the same hour of the day.
- ☐ A MITM attacker could feasibly brute-force the symmetric keys if they were not switched every hour, but could not brute-force the keys if they only had one hour per key.
- ☐ This scheme has no practical purpose because TLS connections cannot last more than an hour, or else the TCP sequence numbers would start being reused.
- ☐ None of the above

Q9.2 (1 point) Suppose the client and server's system clocks are out of sync, and the client uses an old key to send a message to the server, which is using a newer key.

Assume the server has discarded the old key, and is only using the newer key.

What will the server do with this message?

- ○ Accept the ciphertext, and decrypt it to the correct plaintext.
- ○ Accept the ciphertext, and decrypt it to garbage.
- ○ Reject the message because the MAC is invalid.
- ○ Reject the message because the underlying IP packets will get dropped.

In each of the next three subparts, a scheme for switching keys is provided.

An on-path attacker has observed the entire TLS connection so far (starting from the handshake).

The attacker wants to learn K_t, the current key in the connection. Can the attacker learn K_t, and if so, do they need to know K_{t-1}, the previous key in the connection?

Clarification during exam: For 9.4 and 9.5, $K_0 = \mathsf{HKDF}(\texttt{PremasterSecret}, \texttt{"start"})$.

Q9.3 (1 point) To compute the new key, the client and the server compute:

$$\mathsf{HMAC}(\texttt{PremasterSecret}, t)$$

where t is the number of hours elapsed since the beginning of the connection, rounded down.

- ○ The attacker can learn K_t, even if they don't know K_{t-1}.
- ○ The attacker can learn K_t, but only if they know K_{t-1}.
- ○ The attacker cannot learn K_t, even if they know K_{t-1}.

Q9.4 (1 point) To compute the new key, the client and the server compute:

$$\mathsf{HMAC}(\texttt{ClientRandom}, K_{t-1})$$

- ○ The attacker can learn K_t, even if they don't know K_{t-1}.
- ○ The attacker can learn K_t, but only if they know K_{t-1}.
- ○ The attacker cannot learn K_t, even if they know K_{t-1}.

Q9.5 (1 point) To compute the new key, the client and the server compute:

$$\mathsf{HMAC}(\texttt{PremasterSecret}, K_{t-1})$$

- ○ The attacker can learn K_t, even if they don't know K_{t-1}.
- ○ The attacker can learn K_t, but only if they know K_{t-1}.
- ○ The attacker cannot learn K_t, even if they know K_{t-1}.

In each of the next two subparts, a scheme for switching keys is provided.

An on-path attacker and a MITM attacker have observed the entire TLS connection so far (starting from the handshake). Select all true statements.

Q9.6 (2 points) To compute the new key, the client chooses a new, random $\texttt{PremasterSecret}_t$, encrypts it with the server's public key (not any symmetric keys), and sends the encrypted $\texttt{PremasterSecret}_t$ to the server.

K_t is derived from the new $\texttt{PremasterSecret}_t$, the original `ClientRandom`, and the original `ServerRandom`.

- ☐ A on-path attacker can learn K_t, even if they don't know K_{t-1}.
- ☐ A on-path attacker can learn K_t, but only if they know K_{t-1}.
- ☐ A MITM attacker can trick the server into accepting a new key known by the attacker, even if they don't know K_{t-1}.
- ☐ A MITM attacker can trick the server into accepting a new key known by the attacker, but only if they know K_{t-1}.
- ☐ None of the above

Q9.7 (2 points) To compute K_t, the client and server perform a Diffie-Hellman key exchange, with the server signing its half of the exchange. (The exchange is not encrypted with any symmetric keys.)

K_t is derived from the new Diffie-Hellman shared secret, the original `ClientRandom`, and the original `ServerRandom`.

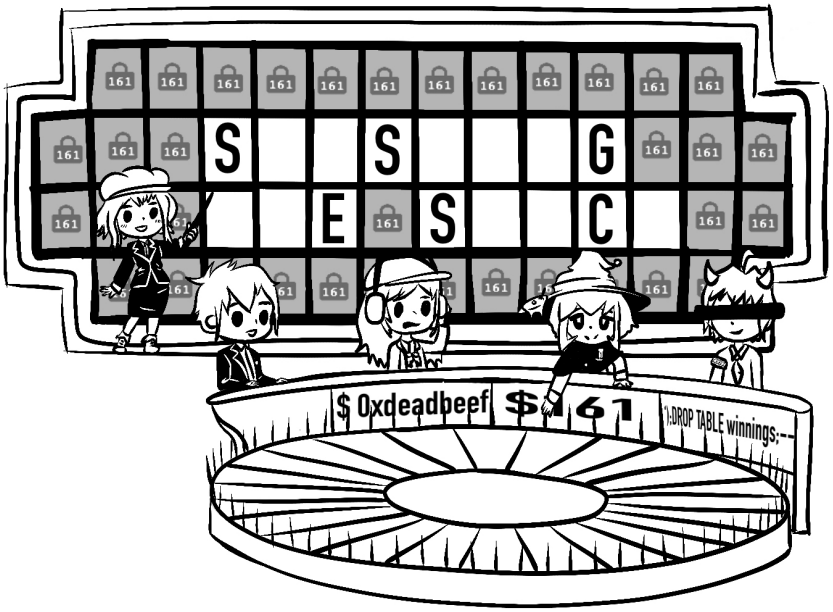
- ☐ A on-path attacker can learn K_t, even if they don't know K_{t-1}.
- ☐ A on-path attacker can learn K_t, but only if they know K_{t-1}.
- ☐ A MITM attacker can trick the server into accepting a new key known by the attacker, even if they don't know K_{t-1}.
- ☐ A MITM attacker can trick the server into accepting a new key known by the attacker, but only if they know K_{t-1}.
- ☐ None of the above

Nothing on this page will affect your grade.

Post-Exam Activity: Wheel of Fortune

Bot, I'd like to solve the puzzle:

Category: Memory Safety



Comment Box

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