Popa and Weaver
Fall 2021

CS 161
Computer Security

Final Exam

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

PRINT your student ID: ____

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

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).

It's EvanBot versus the Caltopian Space Agency! Who do you decide to align with?

This activity will not affect your grade in any way.

EvanBot ○

CSA ○

Q1 *Honor Code* (2 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 and a corresponding notch on Nick's Stanley Fubar demolition tool.

SIGN your name: ____

Q2 *True/false* **(28 points)**

Each true/false is worth 2 points.

Q2.1 True or False: Alice and Bob meet in-person to agree on a shared key K. After they separate, they use K to secure their communications using common cryptographic primitives. This is an example of security through obscurity.

○ True ○ False

Q2.2 A company hires an intern responsible for downloading and analyzing data from a database. The company grants this intern read/write access to the database containing relevant data.

True or False: This is a violation of the principle of least privilege.

○ True ○ False

Q2.3 True or False: A secure PRNG is initialized with a high-entropy seed, then used to generate pseudorandom bits. It is computationally easy for an attacker who sees the pseudorandom bits to learn the seed.

○ True ○ False

Q2.4 True or False: In Bitcoin, if someone has 51% compute power, they can forge transactions.

○ True ○ False

Q2.5 True or False: The same origin-policy (SOP) determines which cookies a browser will include in an HTTP request.

○ True ○ False

Q2.6 True or False: If Mallory successfully executes a DHCP spoofing attack, Alice can no longer rely on the end-to-end security guarantees provided by TLS.

○ True ○ False

Q2.7 True or False: UDP guarantees that packets which are delivered are delivered in order, though it does not guarantee that any given packet is actually delivered.

○ True ○ False

Q2.8 True or False: The TLS layer is responsible for retransmitting TLS packets if they get dropped.

○ True ○ False

Q2.9 True or False: RSA TLS and Diffie-Hellman TLS would both still be safe from replay attacks, even if all clients used the same, fixed value as the client random.

○ True ○ False

Q2.10 True or False: RSA TLS provides the property of forward secrecy, but Diffie-Hellman TLS does not.

○ True ○ False

Q2.11 TRUE or FALSE: If MD5 is used to hash the domain names, then domain enumeration is possible in NSEC3.

○ TRUE ○ FALSE

Q2.12 TRUE or FALSE: Specification-based detection is easier to manage than signature-based detection, because you can easily reuse specifications collected from the security community.

○ TRUE ○ FALSE

Q2.13 TRUE or FALSE: One method of determining if some code is malware is running it in a sandbox (isolated environment) and detecting malicious behavior.

○ TRUE ○ FALSE

Q2.14 TRUE or FALSE: After Alice chooses three nodes in the Tor network to form a circuit, it is possible to establish a connection to all three nodes in parallel.

○ TRUE ○ FALSE

Q2.15 (0 points) `EvanBot is a real bot.`

○ TRUE ○ FALSE

Q3 *Echo, Echo, Echo* **(16 points)**

Consider the following vulnerable C code:

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 
 4 char name[32];
 5 
 6 void echo(void) {
 7     char echo_str[16];
 8     printf("What do you want me to echo back?\n");
 9     gets(echo_str);
10     printf("%s\n", echo_str);
11 }
12 
13 int main(void) {
14     printf("What's your name?\n");
15     fread(name, 1, 32, stdin);
16     printf("Hi %s\n", name);
17 
18     while (1) {
19         echo();
20     }
21 
22     return 0;
23 }
```

Assume you are on a little-endian 32-bit x86 system. Assume that there is no compiler padding or additional saved registers in all questions. For the first 4 parts, assume that **no memory safety defenses** are enabled.

Q3.1 (2 points) Assume that execution has reached line 8. Fill in the following stack diagram. Assume that each row represents 4 bytes.

Stack
1
2
RIP of `echo`
SFP of `echo`
3
4

○ (1) - RIP of `main`; (2) - SFP of `main`; (3) - `echo_str[0]`; (4) - `echo_str[4]`

○ (1) - SFP of `main`; (2) - RIP of `main`; (3) - `echo_str[0]`; (4) - `echo_str[4]`

○ (1) - RIP of `main`; (2) - SFP of `main`; (3) - `echo_str[12]`; (4) - `echo_str[8]`

Q3.2 (3 points) Using GDB, you find that the address of the RIP of `echo` is `0x9ff61fc4`.

Construct an input to `gets` that would cause the program to execute malicious shellcode. Write your answer in Python 2 syntax (like in Project 1). You may reference `SHELLCODE` as a 16-byte shellcode.

Q3.3 (4 points) Which of the following defenses on their own would prevent an attacker from executing the exploit above? Select all that apply.

☐ Stack canaries

☐ ASLR

☐ Pointer authentication

☐ None of the above

☐ Non-executable pages

Q3.4 (5 points) Assume that non-executable pages are enabled so we cannot execute SHELLCODE on stack. We would like to exploit the `system(char *command)` function to start a shell. This function executes the string pointed to by `command` as a shell command. For example, `system("ls")` will list files in the current directory.

Construct an input to `gets` that would cause the program to execute the function call `system("sh")`. Assume that the address of `system` is `0xdeadbeef` and that the address of the RIP of `echo` is `0x9ff61fc4`. Write your answer in Python 2 syntax (like in Project 1).

Hint: Recall that a return-to-libc attack relies on setting up the stack so that, when the program pops off and jumps to the RIP, the stack is set up in a way that looks like the function was called with a particular argument.

Q3.5 (2 points) Assume that, in addition to non-executable pages, ASLR is also enabled. However, addresses of global variables are not randomized.

Is it still possible to exploit this program and execute malicious shellcode?

◯ Yes, because you can find the address of both `name` and `system`

◯ Yes, because ASLR preserves the relative ordering of items on the stack

◯ No, because non-executable pages means that you can't start a shell

◯ No, because ASLR will randomize the code section of memory

Q4 *Challenge-Response* (28 points)

Clarification during exam: Weak unforgeability and unforgeability mean that the attacker is trying to directly authenticate to Bob, while Alice is offline.

Suppose that Alice wants to authenticate herself to Bob, such that Bob knows that he is talking to Alice. Formally, Bob wants to verify that Alice knows some fixed secret K (which Bob may not necessarily know). One common method of authentication is *challenge-response authentication*, which has two main phases:

- **Challenge**: Bob generates a challenge C, and sends C to Alice. C may be different each time Alice wants to authenticate herself.
- **Response**: Alice uses the challenge C and her secret K in order to construct a response R, and she sends R to Bob. If Bob successfully verifies the response, Bob knows that he is talking to Alice.

Consider the following properties of challenge-response authentication:

- **Weak unforgeabilty**: An adversary that does not know K should not be able to authenticate to Bob. Assume that the adversary **has not** observed any past authentication rounds between Alice and Bob.
- **Unforgeabilty**: An adversary that does not know K should not be able to authenticate to Bob. Assume that the adversary **has** observed (but not been a MITM in) past, successful authentication rounds between Alice and Bob.
- **Weak key secrecy**: A **passive** adversary that observes any number of authentication rounds between Alice and Bob cannot recover K.
- **Key secrecy**: An **active** adversary that is a MITM for any number of authentication rounds between Alice and Bob cannot recover K.

For each challenge-response authentication scheme, select the property or properties that the scheme possesses. Assume that:

- Enc/Dec is a semantically secure asymmetric encryption scheme.
- Sign/Verify is a secure, unforgeable digital signature scheme.

Q4.1 (4 points) **Scheme 1**: K is a 128-bit random key known to both Alice and Bob.

Challenge: Bob generates a new asymmetric key pair (SK_B, PK_B) and sends the challenge $C = PK_B$.

Response: Alice sends the response $R = \mathsf{Enc}(PK_B, K)$. Bob verifies that $\mathsf{Dec}(SK_B, R)$ is equal to K.

☐ Weak unforgeabilty ☐ Weak key secrecy ☐ None of the above

☐ Unforgeabilty ☐ Key secrecy

Q4.2 (4 points) **Scheme 2**: $K = SK_A$ is a private key known only to Alice. Bob knows Alice's public key PK_A.

Challenge: Bob sends a challenge $C = r$, where r is a randomly chosen, 256-bit number.

Response: Alice sends the response $R = \mathsf{Sign}(SK_A, C)$. Bob verifies that $\mathsf{Verify}(PK_A, R)$ returns true.

☐ Weak unforgeabilty ☐ Weak key secrecy ☐ None of the above

☐ Unforgeabilty ☐ Key secrecy

Q4.3 (4 points) **Scheme 3**: K is a 128-bit random key known to both Alice and Bob.

Challenge: Bob sends a challenge $C =$ "" (empty message).

Response: Alice sends the response $R = K$. Bob verifies that the response is equal to K.

☐ Weak unforgeabilty ☐ Weak key secrecy ☐ None of the above

☐ Unforgeabilty ☐ Key secrecy

Q4.4 (4 points) **Scheme 4**: $K = SK_A$ is a private key known only to Alice. Bob knows Alice's public key PK_A.

Challenge: Bob sends a challenge $C = \mathsf{Enc}(PK_A, r)$, where r is a randomly chosen, 128-bit number.

Response: Alice sends the response $R = \mathsf{Dec}(SK_A, C)$. Bob verifies that R is equal to r.

☐ Weak unforgeabilty ☐ Weak key secrecy ☐ None of the above

☐ Unforgeabilty ☐ Key secrecy

Q4.5 (4 points) **Scheme 5**: K is a 128-bit random key known to both Alice and Bob.

Challenge: Bob sends a challenge $C =$ "" (empty message).

Response: Alice sends the response $R = \mathsf{H}(K)$, where H is a secure, cryptographic hash function. Bob verifies that R is equal to $\mathsf{H}(K)$.

☐ Weak unforgeabilty ☐ Weak key secrecy ☐ None of the above

☐ Unforgeabilty ☐ Key secrecy

Q4.6 (5 points) Assume that Alice and Bob perform an ephemeral Diffie-Hellman key exchange to derive a symmetric key and form an encrypted channel. Alice authenticates herself to Bob **through** the encrypted channel, using one of the schemes above. Afterwards, Bob receives the message "Send $10 to Charlie," also through the encrypted channel. For which of these schemes can Bob be sure that Alice sent this message? Select all that apply.

☐ Scheme 1 ☐ Scheme 4

☐ Scheme 2 ☐ Scheme 5

☐ Scheme 3 ☐ None of the above

Q4.7 (3 points) Notice that password-based authentication is very similar to Scheme 3: The user uses their password as the secret K and sends it to the server to authenticate themselves. Suppose that Alice authenticates herself their username and password to a bank website that uses secure HTTPS, using Diffie Hellman TLS.

In the same HTTPS channel, Alice makes another request to the bank: "Send $10 to Charlie." Can the bank be sure that Alice made this request? Assume that no one else knows Alice's password. Briefly justify your answer (1–2 sentences).

Q5 *Bob's Birthday* (11 points)

It's Bob's birthday! Alice wants to send an encrypted birthday message to Bob using ElGamal.

Recall the definition of ElGamal encryption:

- b is the private key, and $B = g^b \bmod p$ is the public key.
- $\mathsf{Enc}(B, M) = (C_1, C_2)$, where $C_1 = g^r \bmod p$ and $C_2 = M \times B^r \bmod p$
- $\mathsf{Dec}(b, C_1, C_2) = C_1^{-b} \times C_2 \bmod p$

Q5.1 (2 points) Mallory wants to tamper with Alice's message to Bob. In response, Alice decides to sign her message with an RSA digital signature. Bob receives the signed message and verifies the signature successfully. Can he be sure the message is from Alice?

○ Yes, because RSA digital signatures are unforgeable.

○ Yes, because RSA encryption is IND-CPA secure.

○ No, because Mallory could have blocked Alice's message and replaced it with a different one.

○ No, because Mallory could find a different message with the same hash as Alice's original message.

As we discussed in class, ElGamal is malleable, meaning that a man-in-the-middle can change a message in a *predictable* manner, such as producing the ciphertext of the message $2 \times M$ given the ciphertext of M.

Q5.2 (3 points) Consider the following modification to ElGamal: Encrypt as normal, but further encrypt portions of the ciphertext with a block cipher E, which has a block size equal to the number of bits in p. In this scheme, Alice and Bob share a symmetric key K_{sym} known to no one else.

Under this modified scheme, C_1 is computed as $E_{K_{\mathsf{sym}}}(g^r \bmod p)$ and C_2 is computed as $E_{K_{\mathsf{sym}}}(M \times B^r \bmod p)$. Is this scheme still malleable?

○ Yes, because block ciphers are not IND-CPA secure encryption schemes

○ Yes, because the adversary can still forge $k \times C_2$ to produce $k \times M$

○ No, because block ciphers are a pseudorandom permutation

○ No, because the adversary isn't able to learn anything about the message M

The remaining parts are independent of the previous part.

For Bob's birthday, Mallory hacks into Bob's computer, which stores Bob's private key b. She isn't able to read b or overwrite b with an arbitrary value, but she can multiply the stored value of b by a random value z known to Mallory.

Mallory wants to send a message to Bob that appears to decrypt as normal, but **using the modified key** $b \cdot z$. Give a new encryption formula for C_1 and C_2 that Mallory should use. Make sure you only use values known to Mallory!

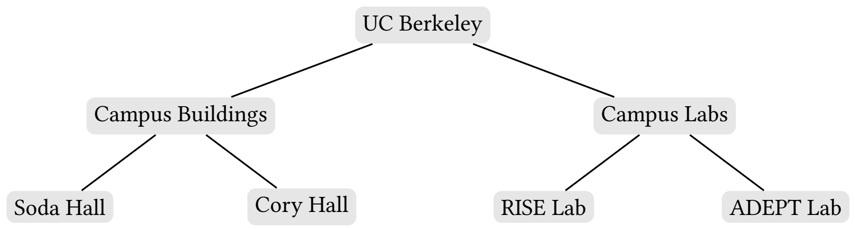
Clarification during exam: For subparts 5.3/5.4, assume that the value of B is unchanged.

Q5.3 (3 points) Give a formula to produce C_1, encrypting M.

Q5.4 (3 points) Give a formula to produce C_2, encrypting M.

Q6 *RISELab Shenanigans* (16 points)

Certificate authorities of UC Berkeley are organized in a hierarchy as follows:



Alice is a student in RISELab at UC Berkeley and wants to obtain a certificate for her public key. Assume that only RISELab is allowed to issue certificates to Alice.

Q6.1 (2 points) Which of the following values are included in the certificate issued to Alice? Select all that apply.

☐ Alice's public key

☐ Alice's private key

☐ A signature on Alice's *public* key, signed by RISELab's private key

☐ A signature on Alice's *private* key, signed by RISELab's private key

☐ None of the above

Q6.2 (2 points) Assume that the only public key you trust is UC Berkeley's public key. Which certificates do you need to verify in order to be sure that you have Alice's public key? Select all that apply.

☐ Certificate for Alice

☐ Certificate for Soda Hall

☐ Certificate for RISELab

☐ Certificate for Campus Labs

☐ None of the above

Q6.3 (4 points) RISELab issues a certificate to Alice that expires in 1 hour. Which of the following statements are true about using such a short expiration date? Select all that apply.

☐ It mitigates attacks where Alice's private key is stolen

☐ It mitigates attacks where RISELab's private key is stolen

☐ It mitigates attacks where Campus Labs' private key is stolen

☐ It forces Alice to renew the certificate more often

☐ None of the above

The following subparts are independent from the previous subparts.

Passwords on the RISELab website are six-digit codes, where each digit is one of 0–9 (repeat digits are allowed). An attacker steals the password database, which includes Alice's hashed password, and wants to learn Alice's password.

For each password storage scheme, *in the worst case*, how much time would it take for the attacker to brute-force Alice's password?

Assumptions:

- The attacker tries passwords one at a time.
- H is a hash function that takes 1 second to compute.
- The time required for all other operations is negligible.

Q6.4 (2 points) Passwords are stored as $H(\mathsf{pwd})$.

○ $10^6 \cdot 2 \cdot 8$ seconds ○ $10^6 \cdot 2^8$ seconds ○ 2^8 seconds

○ $6 \cdot 10 \cdot 2^8$ seconds ○ 10^6 seconds

Q6.5 (2 points) Passwords are stored as $(\mathsf{salt}, H(\mathsf{salt} \| \mathsf{pwd}))$, where salt is an 8-bit random string.

○ $10^6 \cdot 2 \cdot 8$ seconds ○ $10^6 \cdot 2^8$ seconds ○ 2^8 seconds

○ $6 \cdot 10 \cdot 2^8$ seconds ○ 10^6 seconds

Q6.6 (4 points) Assume that the attacker is conducting an **online** brute-force attack against Alice's account. Which of the following changes, if implemented individually, would make it more difficult for the attacker to access Alice's account? Select all that apply.

☐ Alice uses a random, alphanumeric, 32-character password instead of a 6-digit numeric password

☐ Alice enables two-factor authentication on her account

☐ RISELab imposes a timeout which doubles after each password attempt

☐ RISELab enables TLS for its login page

☐ None of the above

Q7 *SQL Injection* (20 points)

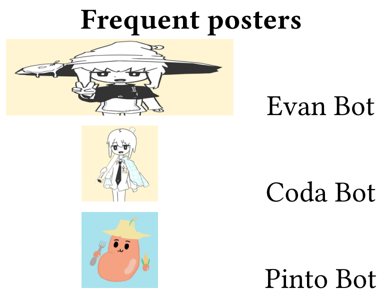
CS 161 students are using a modified version of Piazza to discuss project questions! In this version, the names and profile pictures of the students who answer questions frequently are listed on a side panel on the website.

The server stores a table of users with the following schema:

```
CREATE TABLE users (
    First TEXT,                  -- First name of the user.
    Last TEXT,                   -- Last name of the user.
    ProfilePicture TEXT,         -- URL of the image.
    FrequentPoster BOOLEAN,      -- Are they a frequent poster?
);
```

Q7.1 (3 points) Assume that you are a frequent poster. When playing around with your account, you notice that you can set your profile picture URL to the following, and your image on the frequent poster panel grows wider than everyone else's photos:

ProfilePicture URL: `https://cs161.org/evan.jpg" width="1000`



What kind of vulnerability might this indicate on Piazza's website?

○ Stored XSS ○ Path traversal attack

○ Reflected XSS ○ Buffer overflow

○ CSRF

Q7.2 (3 points) Provide a malicious image URL that causes the JavaScript `alert(1)` to run for any browser that loads the frequent poster panel. Assume all relevant defenses are disabled.

Hint: Recall that image tags are typically formatted as `<img src="image.png">`.

Q7.3 (4 points) Suppose your account is not frequent poster, but you still want to conduct an attack through the frequent posters panel!

When a user creates an account on Piazza, the server runs the following code:

```
query := fmt.Sprintf("
    INSERT INTO users (First, Last, ProfilePicture, FrequentPoster)
        VALUES ('%s', '%s', '%s', FALSE);
    ",
    first, last, profilePicture)
db.Exec(query)
```

Provide an input for `profilePicture` that would cause your malicious script to run the next time a user loads the frequent posters panel. You may reference `PAYLOAD` as your malicious image URL from earlier, and you may include `PAYLOAD` as part of a larger input.

Q7.4 (4 points) Instead of injecting a malicious script, you want to conduct a DoS attack on Piazza! Provide an input for `profilePicture` that would cause the SQL statement `DROP TABLE users` to be executed by the server.

Suppose that session cookies are used to authenticate to Piazza. This token is checked whenever the user sends a request to Piazza.

Clarification during exam: "Your malicious script" refers to your exploit in 7.2.

Q7.5 (3 points) Your malicious script submits a GET request to the Piazza website that marks "helpful!" on one of your comments. Does the same-origin policy defend against this attack?

- ○ Yes, because the same-origin policy prevents the script from making the request
- ○ Yes, because the script runs with the origin of the attacker's website
- ○ No, because the same-origin policy does not block any requests from being made
- ○ No, because the script runs with the origin of Piazza's website

Q7.6 (3 points) Your malicious script submits a GET request to the Piazza website that marks "helpful!" on one of your comments. Does enabling CSRF tokens defend against this attack?

◯ Yes, because the attacker does not know the value of the CSRF token

◯ Yes, because the script runs with the origin of the attacker's website

◯ No, because GET requests do not change the state of the server

◯ No, because the script runs with the origin of Piazza's website

Q8 *Integration Testing* (28 points)

Note: We think this is the hardest question on the exam. Feel free to skip it for now and come back to it later.

Engineers at Gradescope have introduced a new scheme for managing session tokens. Here's how it works:

1. Alice visits `https://gradescope.com/login`, enters her email/password, and clicks "Submit".
2. The JavaScript on the login page makes a POST request to `https://gradescope.com/login` with Alice's email and password.
3. The server validates Alice's email and password and sends a randomly generated session token in the **response body** (not as a cookie).
4. The JavaScript on the login page extracts the token from the response and redirects to `https://gradescope.com/home?token=[sessiontoken]`

The Gradescope web server will check that any URL that requires the user to be signed in contains the `token=[sessiontoken]` parameter in the query string. If it is not present, the server will return an "Access Denied" page instead of responding with the page itself. If successful, the page will also contain the session token somewhere on the page:

```
<p id="token">[sessiontoken]</p>
```

The server ensures that all future requests contain the same token by including `?token=[sessiontoken]` in all Gradesscope links that are present in the HTML response. For example, the link to go to the `https://gradescope.com/grades` page will have its URL set to `https://gradescope.com/grades?token=[sessiontoken]`.

For all parts of this question, make the following assumptions:

- All TLS connections use Diffie-Hellman TLS.
- Alice's browser always sends a Referer header.
- The attacker has full control over the `evil.com` web server.
- All parts of this question are independent.

Q8.1 (5 points) After Alice completes the sign-in flow, she sees a number of links on the homepage she can click on. **An attacker has discovered a vulnerability on each linked page,** allowing them to execute arbitrary JavaScript on that page. If Alice clicks on these links, which of the following links may cause her session token to be leaked to the attacker? Select all that apply.

☐ `http://gradescope.com/assignments?token=[sessiontoken]`

☐ `https://gradescope.com:1234/?token=[sessiontoken]`

☐ `http://support.gradescope.com/?token=[sessiontoken]`

☐ `https://gradescope.com/assignments?token=[sessiontoken]`

☐ `http://evil.com/`

☐ None of the above

Q8.2 (5 points) After Alice completes the sign-in flow, she clicks on the following link from the homepage:

```
https://gradescope.com/assignments?token=[sessiontoken]
```

Eve, an on-path eavesdropper, observes a snapshot of the request and response corresponding to this query. Which of the following attackers could individually collude with Eve for Eve to be able to learn Alice's **session token**? Select all that apply.

☐ An attacker with control over the Gradescope web server

☐ An attacker who's capable of solving the discrete-log problem

☐ An attacker who's capable of exerting infinite computational power

☐ An attacker who has full control over a HIDS on Alice's device

☐ An attacker who can see Alice's cookies

☐ None of the above

Q8.3 (3 points) After Alice completes the sign-in flow, she clicks on the following link from the homepage:

```
https://gradescope.com/assignments?token=[sessiontoken] (Version 1)
http://gradescope.com/assignments?token=[sessiontoken] (Version 2)
```

For this question part only, assume that Alice uses the Tor browser instead of her regular browser. Which of the compromised nodes could learn Alice's **session token**? Select all that apply.

☐ A compromised exit node in a two-node circuit

☐ A compromised exit node in a three-node circuit

☐ The node in a one-node circuit

☐ A compromised entry node in a two-node circuit

☐ A compromised middle node in a three-node circuit

☐ None of the above

Q8.4 (5 points) After Alice completes the sign-in flow, she sees a number of links on the homepage she can click on. An **on-path** attacker wants to set a cookie with the following attributes in Alice's browser:

```
Domain=gradescope.com
Path=/home
HttpOnly=true
```

If Alice clicks on these links, which of the following links would allow the attacker to set a cookie with these attributes in Alice's browser? Select all that apply.

☐ `http://gradescope.berkeley.edu/`

☐ `http://gradescope.com:8081/?token=[sessiontoken]`

☐ `http://support.gradescope.com/grades?token=[sessiontoken]`

☐ `http://evil.com`

☐ `https://gradescope.com/?token=[sessiontoken]`

☐ None of the above

Q8.5 (5 points) After Alice completes the sign-in flow, she sees a number of links on the homepage she can click on. An attacker **who has control over the web server at each of these links** wants to set a cookie with the following attributes in Alice's browser:

```
Domain=gradescope.com
Path=/home
Secure=true
```

If Alice clicks on these links, which of the following links may allow the attacker to set a cookie with these attributes in Alice's browser? Select all that apply.

☐ `http://gradescope.com/?token=[sessiontoken]`

☐ `https://gradescope.berkeley.edu`

☐ `https://support.gradescope.com/grades/?token=[sessiontoken]`

☐ `https://evil.com/`

☐ `https://gradescope.com/?token=[sessiontoken]`

☐ None of the above

Q8.6 (5 points) Gradescope uses a `language` cookie with the following attributes:

```
Domain=gradescope.com
Path=/grades
Secure=false
HttpOnly=true
```

After Alice completes the sign-in flow, she sees a number of links on the homepage she can click on. If Alice clicks on these links, which of the following links may allow the attacker to learn the cookie value?

Assume that the attacker can observe and modify the request/response information of **only** the request made by Alice clicking the link and nothing else. Select all that apply.

☐ `http://support.gradescope.com/grades?token=[sessiontoken]`

☐ `http://berkeley.edu/grades/1234/`

☐ `http://gradescope.com/grades/?token=[sessiontoken]`

☐ `http://gradescope.cs161.org/home/`

☐ `http://evil.com/`

☐ None of the above

Q9 *Pancake Query Protocol* (19 points)

EvanBot is already prepared for the winter break but realizes that there are no more pancakes and needs to order more! To speed up the ordering process, EvanBot crafts a custom Pancake Query Protocol (PQP) and needs to ensure that it is secure.

PQP runs directly over IP, and a PQP packet contains the following information:

- A packet type
- The pancake query data (either a request for an order, or the order itself)

For now, assume that the only packet type supported by PQP is the ORDER type. For example, EvanBot might send the following PQP packet:

- EvanBot ⟶ Restaurant: {Type: ORDER; Data: "I want 1 stack of blueberry pancakes!"}

For all parts, assume that EvanBot knows the IP address of the restaurant. All subparts of this question are independent.

Q9.1 (5 points) Which of the following statements are true about PQP? Select all that apply.

☐ An off-path attacker can learn EvanBot's order

☐ An off-path attacker can trick the restaurant into cooking unwanted pancakes for EvanBot

☐ An on-path attacker can conduct a RST injection attack

☐ An on-path attacker can learn EvanBot's order

☐ EvanBot can be sure that the restaurant received the order

☐ None of the above

Q9.2 (3 points) EvanBot adds an ACK packet type to PQP packets. After a restaurant receives an order, the restaurant sends an ACK packet acknowledging the order. If EvanBot does not receive the ACK, EvanBot re-sends the order until an ACK is received.

EvanBot tries to order 1 stack of pancakes from the restaurant and eventually receives an ACK. Assume that **no** network attackers are present. How many orders could the restaurant receive?

○ Exactly 0, because IP is unreliable.

○ Either 0 or 1, because IP is unreliable.

○ 0 or more, because IP is unreliable.

○ Exactly 1, because the restaurant ACKs any order it receives.

○ 1 or more, because EvanBot might try more than once.

○ 2 or more, because the restaurant may send multiple ACKs.

Q9.3 (4 points) Consider the following modification to PQP: To order, the client generates a random order ID and sends it in the PQP ORDER packet along with the order. The server sends back the order ID in the PQP ACK packet.

Can an off-path attacker trick the restaurant into accepting a spoofed order appearing to come from EvanBot? Briefly justify your answer (1–2 sentences).

○ Yes ○ No

Q9.4 (4 points) Which of the following modifications to PQP, if made individually, would prevent an off-path attacker from tricking the restaurant into accepting spoofed orders? Select all that apply.

☐ The restaurant generates a random order ID and sends it back in the PQP ACK. The restaurant must receive a PQP ACK-ACK packet from EvanBot containing the order ID to confirm the order.

☐ The restaurant sends a fixed time-to-live (TTL) to EvanBot in the PQP ACK. The restaurant must receive a final, empty PQP ACK packet from EvanBot within the TTL to confirm the order.

☐ PQP runs over UDP instead of IP, and EvanBot chooses a random source port.

☐ PQP runs over TCP instead of IP.

☐ None of the above

Q9.5 (3 points) EvanBot proposes an additional packet types for PQP: LISTORDERS. When a restaurant receives an LISTORDERS packet, it responds with a list of all orders that it has ever received from any customer. Name one security issue with this proposal and describe the steps an attacker should take to exploit this issue (1–2 sentence).

Q10 *In Medias Res* (14 points)

Recall that TLS uses a handshake to form a new connection. One possible optimization for TLS is to store the symmetric keys associated with a TLS session and use those keys to "resume" a previous TLS session, instead of performing a new handshake for every connection.

One implementation of session resumption uses a session ticket. During the initial connection, the handshake is as follows:

1. The client sends a ClientHello with no associated ticket.
2. The rest of the standard TLS handshake proceeds as normal.
3. The server generates a **session ticket** and sends the ticket to the client through the encrypted TLS channel.
4. The client stores the ticket.

If the client wants to resume this connection later:

1. The client sends a ClientHello to the server with the ticket from the initial connection.
2. The server uses the ticket to "resume" a TLS connection.

For each scheme for generating and using the session ticket, select all statements that are true about the scheme. Assume that Diffie-Hellman TLS is used and that an on-path attacker has observed and recorded both the initial TLS handshake and a resumed handshake between the client and server.

Q10.1 (3 points) The ticket contains the TLS symmetric keys used in the initial connection. The resumed TLS session uses the symmetric keys present in the ticket.

☐ The attacker can trick the server into resuming a TLS connection that uses any symmetric keys of their choosing

☐ The attacker, if they compromise the server, can decrypt messages from the initial TLS connection

☐ The attacker, if they do not compromise the server, can decrypt messages from the initial TLS connection

☐ None of the above

Q10.2 (3 points) The server randomly generates a *new* set of symmetric keys and stores them in the ticket. The resumed TLS session uses the symmetric keys present in the ticket.

☐ The attacker can trick the server into resuming a TLS connection that uses any symmetric keys of their choosing

☐ The attacker, if they compromise the server, can decrypt messages from the initial TLS connection

☐ The attacker, if they do not compromise the server, can decrypt messages from the initial TLS connection

☐ None of the above

Q10.3 (3 points) The ticket contains the same symmetric keys as the initial connection. The ticket is then encrypted and MAC'd with a **ticket key** known only to the server. The server verifies and decrypts the ticket, and then the resumed TLS session uses the symmetric keys present in the ticket.

☐ The attacker can trick the server into resuming a TLS connection that uses any symmetric keys of their choosing

☐ The attacker, if they compromise the server, can decrypt messages from the initial TLS connection

☐ The attacker, if they do not compromise the server, can decrypt messages from the initial TLS connection

☐ None of the above

Q10.4 (3 points) The ticket contains a session token randomly generated by the server, and the server stores the initial connection's symmetric keys associated with this token. When resuming a TLS session, it uses the received token to look up its associated keys, and resumes the TLS session uses those keys.

☐ The attacker can trick the server into resuming a TLS connection that uses any symmetric keys of their choosing

☐ The attacker, if they compromise the server, can decrypt messages from the initial TLS connection

☐ The attacker, if they do not compromise the server, can decrypt messages from the initial TLS connection

☐ None of the above

Q10.5 (2 points) EvanBot suggests storing parameters from a the initial TLS handshake in the ticket. Which of the following values would need to be included in the ticket for the client and server to be able to re-derive the same symmetric keys from the initial connection? Select all that apply.

☐ Client random

☐ Server random

☐ Premaster secret

☐ The MAC of the dialog

☐ None of the above

Q11 *DNSSEC* (18 points)

EvanBot wants to know the IP address of `security.csa.gov`.

Q11.1 (2 points) EvanBot performs a DNS lookup and receives the following records from the root name server:

#	Name	Type	Value
1	`.gov`	NS	`gov-ns.com`
2	`gov-ns.com`	A	`140.0.0.1`

EvanBot thinks the root name server might be broken, because it incorrectly endorsed `gov-ns.com` as a `.gov` name server. Is EvanBot right?

◯ Yes, because `gov-ns.com` is a `.com` name server

◯ Yes, because `gov-ns.com` is a domain, not a name server

◯ No, because `gov-ns.com` can answer both `.com` and `.gov` queries

◯ No, because the domain of the name server doesn't need to be in the name server's zone

Q11.2 (3 points) Next, EvanBot receives the following records from the `.gov` name server:

#	Name	Type	Value
3	`csa.gov`	NS	`csa-ns.gov`
4	`csa.gov`	NS	`csa-ns.com`
5	`csa-ns.gov`	A	`150.0.0.1`
6	`csa-ns.com`	A	`160.0.0.1`
7	`csa.gov`	A	`170.0.0.1`

If EvanBot performs bailiwick checking, which A records will be stored into EvanBot's cache? Select all that apply.

☐ Record 5

☐ Record 6

☐ Record 7

☐ None of the above

Q11.3 (4 points) Finally, EvanBot receives the following record from the `csa.gov` name server:

#	Name	Type	Value
8	`security.csa.gov`	A	`180.0.0.1`

Under which of these threat models can EvanBot be confident that this answer record is correct and not malicious? Assume that source port randomization is enabled, and EvanBot makes only one request. Assume that the recursive resolver is not malicious. Select all that apply.

☐ There are no network attackers, but a name server might be malicious

☐ No name servers are malicious, but there is a man-in-the-middle attacker present

☐ No name servers are malicious, but there is an on-path attacker present

☐ No name servers are malicious, but there is an off-path attacker present

☐ None of the above

Q11.4 (2 points) EvanBot decides to perform the query with DNSSEC for extra security. EvanBot receives the following DNSSEC records from the root name server:

#	Type	Contents
9	DNSKEY	root's public key
10	DS	H(`.gov`'s public key)
11	RRSIG	Signature on record 10

EvanBot thinks the root name server might be broken. Is EvanBot right?

◯ Yes, because it never provided a signature on `.gov`'s public key

◯ Yes, because it never provided `.gov`'s public key

◯ No, because we can implicitly trust any records coming from the root name server

◯ No, because it provided enough information for us to trust the `.gov` name server

Q11.5 (4 points) EvanBot receives the following DNSSEC records from the `.gov` name server:

#	Type	Contents
12	DNSKEY	`.gov`'s public key
13	DS	H(`csa.gov`'s public key)
14	RRSIG	Signature on record 13
15	A	IP address of `evil.csa.gov`
16	RRSIG	Signature on record 15

EvanBot validates that these records have a valid chain of trust back to root, but suspects that some of these records might have been added by an attacker to poison EvanBot's cache.

Which of the following attackers could have successfully poisoned EvanBot's cache with these records? Select all that apply.

☐ A man-in-the-middle attacker who has not compromised any private keys

☐ An on-path attacker who has not compromised any private keys

☐ A man-in-the-middle attacker who has compromised `.gov`'s private key

☐ An on-path attacker who has compromised `.gov`'s private key

☐ None of the above

Q11.6 (3 points) EvanBot receives the following DNSSEC record from the `csa.gov` name server:

#	Type	Contents
17	RRSIG	Signature on record 8

How many signature verifications does EvanBot need to perform to ensure that the IP address in record 8 is correct?

○ 0 ○ 2 ○ 4

○ 1 ○ 3 ○ More than 4

Doodle Page

Nothing on this page will not affect your grade in any way.

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

Post-Exam Activity