Peyrin & Ryan
Summer 2020

CS 161 Computer Security

Final Exam

For questions with **circular bubbles**, you may select exactly *one* choice on Gradescope.

○ Unselected option

● Only one selected option

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

■ You can select

■ multiple squares

For questions with a **large box**, you need to write a short answer in the corresponding text box on Gradescope.

You have 170 minutes. There are 10 questions of varying credit (250 points total).

The exam is open note. You can use an unlimited number of handwritten cheat sheets, but you must work alone.

Clarifications will be posted at https://cs161.org/clarifications.

Q1 *MANDATORY – Honor Code* (7 points)

Read the honor code on the Gradescope answer sheet and type your name. ***Failure to do so will result in a grade of 0 for this exam.***

Q2 *True/false* (56 points)

Each true/false is worth 2 points.

Q2.1 TRUE or FALSE: You should always use HMAC instead of any other MAC because HMAC has stronger integrity and authentication guarantees than any other MAC.

○ TRUE ● FALSE

Solution: False. All MACs provide the same integrity and authentication guarantees.

Q2.2 TRUE or FALSE: A MiTM during the Diffie-Hellman Key Exchange can force both parties to derive a shared key (that the MiTM doesn't necessarily know) that is different than the one they would've derived otherwise.

● TRUE ○ FALSE

Solution: True. Mallory can modify $g^a \longrightarrow g^{am}$ and $g^b \longrightarrow g^{bm}$, causing both parties to derive the key g^{abm}.

Q2.3 TRUE or FALSE: A MiTM during the Diffie-Hellman Key Exchange can force both parties to unknowingly derive different keys that the MiTM knows.

● TRUE ○ FALSE

Solution: True. This is the standard MiTM attack from lecture.

Q2.4 TRUE or FALSE: A MiTM during the Diffie-Hellman Key Exchange can force both parties to derive a set of pre-determined keys that the MiTM knows.

○ TRUE ● FALSE

Solution: False. The MiTM can force the parties to derive keys that they know, but they cannot predetermine these keys since both parties contribute randomness. For example, if Mallory wants Alice to derive the key y, and is given g^a, she must find x s.t. $(g^a)^x = y$ which would require breaking discrete log.

Q2.5 TRUE or FALSE: CSRF tokens are an effective defense against CSRF attacks only if clients' browsers respect the same-origin policy.

● TRUE ○ FALSE

Solution: True. By SOP, websites on another domain are unable to access the content of the website on the target domain. If browsers did not respect SOP, a malicious website could access the CSRF token in another page.

Q2.6 TRUE or FALSE: An XSS vulnerability in a website cannot be exploited to gain control over a user's session if the session cookie has the HttpOnly flag set.

○ True [✓ ● False]

> **Solution:** False. While the attacker may not be able to actually learn the value of the cookie, the XSS vulnerability still allows the attacker to violate SOP and make malicious requests under the user's session.

Q2.7 True or False: `https://secure.bank.com` is able to set the following cookie using the Set-Cookie header: `session=1234567; Domain=bank.com; HttpOnly`.

[✓ ● True] ○ False

> **Solution:** True. `bank.com` is a prefix of the current domain `secure.bank.com`.

Q2.8 True or False: A user wants their web traffic to appear like it's coming from somewhere else with the lowest latency possible. This user should prefer a VPN instead of Tor.

[✓ ● True] ○ False

> **Solution:** True. If all the user wants is to bounce their location, a VPN will be faster than Tor.

Q2.9 True or False: In Bitcoin, once a transaction is successfully added to the blockchain, it can never be lost.

○ True [✓ ● False]

> **Solution:** False. The blockchain could fork and not include your transaction.

Q2.10 When you log in to Zoom, you make a POST Request to `https://zoom.us/berkeley/signin` with an email and password in the form data. The Response contains a session token cookie **without** the Secure flag set.

True or False: An on-path attacker could steal your session token by observing only this request.

○ True [✓ ● False]

> **Solution:** False. The request is an HTTPS request, which indicates that the username and password are encrypted under TLS.

Q2.11 When you go to `https://berkeley.zoom.us/m/stanford`, you see an image of Stanford's lawn. The page source shows that the image is being loaded from `http://stanford.zoom.us/i/stanford.png`.

True or False: This a violation of the same-origin policy.

○ True [✓ ● False]

Solution: False. The Same-Origin Policy does not restrict sites from loading third-party images.

Q2.12 You're using Tor with three intermediate nodes. Assume all nodes are handling a large amount of traffic.

True or False: Even if two of those nodes are compromised, your anonymity is still protected.

[*Clarification during exam*: This question was thrown out during the exam, and both True and False were accepted as valid answers. See solution for why.]

[✓ ● True] ○ False

Solution: The intended answer was true. Since one of the nodes is honest, the malicious nodes won't be able to link any specific traffic to you.

However, we did not specify if two nodes could collude. If two nodes can collude, they might be able to use timing patterns to link traffic to your identity, depending on how much traffic constitutes "a large amount of traffic."

Because we felt this question was ambiguous, both True and False were accepted as valid answers.

Q2.13 Instead of using Tor, you forward your traffic through three intermediate proxies **unencrypted**. Using these proxies, you log into `https://twitter.com`

True or False: Assuming the entry proxy is honest, the middle and exit proxies cannot figure out your identity

[✓ ● True] ○ False

Solution: True. This proxy does not see your IP address, and since your communication with Twitter is over TLS, the proxy doesn't learn your session cookies, content you're reading/sending, etc.

Q2.14 You decide to use a recursive resolver which uses DNSSEC. Your client uses standard DNS.

True or False: An on-path adversary cannot poison your client's cached DNS records.

○ True [✓ ● False]

Solution: False. An on-path attacker can still do basic DNS spoofing between the resolver and client.

Q2.15 A recursive resolver supports DNSSEC. The resolver contacts three other nameservers to answer a certain query.

TRUE or FALSE: All three nameservers must support DNSSEC in order for DNSSEC to provide any guarantees.

● TRUE ○ FALSE

Solution: True. If any of the nameservers don't support DNSSEC, then the certificate chain will be broken.

Q2.16 TRUE or FALSE: DHCP is secure against an on-path attacker.

○ TRUE ● FALSE

Solution: False. If the on-path attacker sends a fake response before the legitimate response, they can convince the victim to accept an incorrect configuration.

Q2.17 TRUE or FALSE: Using HTTPS is a good defense against clickjacking attacks.

○ TRUE ● FALSE

Solution: False. In a clickjacking attack, the victim is already interacting with a malicious website. Even if the victim was contacting the malicious website securely, the attack would still be possible.

Q2.18 TRUE or FALSE: Spearphishing is more dangerous than standard phishing because it uses information about the victim.

● TRUE ○ FALSE

Solution: True. The victim is more likely to be fooled by a spearphishing attack because it includes information specific to the victim, such as their name.

Q2.19 TRUE or FALSE: If a website only allows HTTPS connections, it is secure from SQL injection attacks.

○ TRUE ● FALSE

Solution: False. HTTPS protects the website against network attackers. The attacker can make a secure connection to the website and inject SQL.

Q2.20 TRUE or FALSE: Parameterized SQL stops all SQL injection attacks.

[✓ TRUE] FALSE

Solution: True. As shown in lecture, parameterized SQL precompiles queries so user input cannot be interpreted as code.

Q2.21 Consider a website which inserts user input into a database using a SQL query. The information in the database is then used in subsequent internal SQL queries.

TRUE or FALSE: If the SQL query that accepts user input is parameterized, but the internal ones do not, then the website will be secure from SQL injection attacks.

TRUE [✓ FALSE]

Solution: The second-order SQL injection as shown in discussion can still occur. User input is sanitized in the query that accepts user input, but not in the internal queries, so user input can still be treated as code in the internal inputs.

Q2.22 TRUE or FALSE: Return-oriented programming (ROP) is not effective if non-executable pages (DEP or W^X) are enabled.

TRUE [✓ FALSE]

Solution: False. ROP relies on existing library code in memory. DEP would make this code read-only, but still executable. The attacker never needs to execute any code that they write into memory.

Q2.23 TRUE or FALSE: Format string vulnerabilites are not effective if ASLR is enabled.

TRUE [✓ FALSE]

Solution: False. Format strings can still leak addresses on the stack which can lead to memory safety exploits.

Suppose you find a stored XSS vulnerability on `https://berkeley.zoom.us/m/1234`.

Q2.24 TRUE or FALSE: Some cookies set by `https://berkeley.zoom.us/` could be **read** using your exploit.

[✓ TRUE] FALSE

Solution: True. Any cookies with the HttpOnly flag set to FALSE would be readable by this XSS exploit.

Q2.25 TRUE or FALSE: Some cookies set by `https://berkeley.zoom.us/` could be **modified** using your exploit.

[✓ True] False

Solution: True. XSS would allow you to overwrite any cookies in the appropriate scope.

Q2.26 True or False: Some cookies set by `http://zoom.berkeley.edu/m/1234` could be **read** using your exploit.

True [✓ False]

Solution: False. zoom.berkeley.edu would only be able to set cookies for Cookie-Domain=zoom.berkeley.edu or Cookie-Domain=berkeley.edu - neither of which are accessible via the site with our reflected XSS attack.

Q2.27 True or False: Some cookies set by `https://berkeley.zoom.us/m/1234` could be **modified** using your exploit.

[✓ True] False

Solution: True. JavaScript code executed from a site can always set arbitrary cookies for that site.

Q2.28 True or False: Some cookies set by `http://stanford.zoom.us/m/1234` could be **read** using your exploit.

[✓ True] False

Solution: True. Any cookies with the domain `.zoom.us` and the HttpOnly flag set to FALSE would be readable by JavaScript run from `berkeley.zoom.us`.

This is the end of Q2. Proceed to Q3 on your answer sheet.

Q3 *Password Storage* (28 points)

Bob is trying out different methods to securely store users' login passwords for his website.

Mallory is an attacker who can do some amount of *offline* computation before she steals the passwords file, and some amount of *online* computation after stealing the passwords file.

Technical details:

- Each user has a unique username, but several users may have the same password.
- Mallory knows the list of users registered on Bob's site.
- Bob has at most 500 users using his website with passwords between 8–12 letters.
- Mallory's dictionary contains all words that are less than 13 letters. [*Clarification during exam*: Mallory's dictionary contains all possible user passwords.]
- Mallory can do N online computations and $500N$ offline computations where N is the number of words in the dictionary.
- Slow hash functions take 500 computations per hash while fast hash functions require only 1 computation.[1]

Notation:

- $\mathsf{H_S}$ and $\mathsf{H_F}$, a slow and fast hash function
- Sign, a secure signing algorithm
- uname and pwd, a user's username and password
- k, a signing key known only by Bob

If Bob decides to use signatures in his scheme, assume he will verify them when processing a log-in.

Q3.1 (2 points) How many times could Mallory hash every word in the dictionary using $\mathsf{H_S}$ with **offline computation**?

(A) She can't hash the whole dictionary
[✓ (B) 1]
(C) 500
(D) None of the above
(E) —
(F) —

> **Solution:** Since evaluating a slow hash function takes 500 computations, hashing the entire dictionary will take 500N computations which is the exact amount of offline computation Mallory has

Q3.2 (2 points) How many times could Mallory hash every word in the dictionary using $\mathsf{H_F}$ with **online computation**?

(G) She can't hash the whole dictionary
[✓ (H) 1]
(I) 500
(J) None of the above
(K) —
(L) —

[1]Keep in mind this is much faster than a real-life slow hash function.

Solution: Since evaluating a fast hash function takes 1 computation, hashing the entire dictionary will take N computations which is the exact amount of online computation Mallory has

Q3.3 (2 points) How many times could Mallory hash every word in the dictionary using $\mathsf{H_S}$ with **online computation**?

[✓ (A) She can't hash the whole dictionary]

○ (B) 1

○ (C) 500

○ (D) None of the above

○ (E) ——

○ (F) ——

Solution: As before, hashing the whole dictionary with the slow hash function takes 500N computation but Mallory only has N online computation. Thus, she can't has the whole dictionary

For each part below, indicate all of the things Mallory can do given the password storage scheme. Assume Mallory knows each scheme. **Unless otherwise specified, assume that she can use both offline and online computation**

Q3.4 (4 points) Each user's password is stored as $\mathsf{H_F}(\mathsf{pwd} \| \texttt{'Bob'})$.

[✓ (G) Learn whether two users have the same password with only online computation]

[✓ (H) Learn a specific user's password]

[✓ (I) Change a user's password without detection]

[✓ (J) Learn every user's password]

☐ (K) None of the above

☐ (L) ——

Solution: Since this is a hash function with the same salt, Mallory can do one full run through of the dictionary with online computation to learn each user's password. Additionally, there are no authenticity checks so Mallory can edit a password.

Q3.5 (4 points) Each user's password is stored as the tuple $(\mathsf{H_S}(\mathsf{pwd} \| \texttt{'Bob'}), \mathsf{Sign}(\mathsf{k}, \mathsf{H_F}(\mathsf{pwd})))$.

[✓ (A) Learn whether two users have the same password with only online computation]

[✓ (B) Learn a specific user's password]

[✓ (C) Change a user's password without detection]

[✓ (D) Learn every user's password]

☐ (E) None of the above

☐ (F) ——

Solution: Because of the slow hash, Mallory can only longer do a full run through of the dictionary using online computation. However, she can do so using offline computation since the salt is the same for all passwords. Since the signature does not include the username, password entries can be swapped without detection.

An earlier version of the solutions incorrectly marked (A) as incorrect. However, since signatures are unsalted, an attacker can learn if two users have the same password by comparing signatures (which requires no computation).

Q3.6 (4 points) Each user's password is stored as the tuple $(\mathsf{H_F}(\mathsf{pwd} \,\|\, \mathsf{uname}), \mathsf{Sign}(\mathsf{k}, \mathsf{uname} \,\|\, \mathsf{H_F}(\mathsf{pwd})))$

☐ (G) Learn whether two users have the same password with only online computation

■ (H) Learn a specific user's password

☐ (I) Change a user's password without detection

■ (J) Learn every user's password

☐ (K) None of the above

☐ (L) ——

Solution: Because the salt is now different, Mallory only has enough online computation to bruteforce a single password. However, using offline computation she can still learn all the passwords since she can bruteforce the dictionary 500 times. Since each signature is tied to a specific user and Mallory doesn't know k, she can't edit a user's password.

Q3.7 (4 points) Each user's password is stored as $(\mathsf{H_S}(\mathsf{pwd} \,\|\, \mathsf{uname}), \mathsf{Sign}(\mathsf{k}, \mathsf{H_S}(\mathsf{pwd})))$

[*Clarification during exam*: The expression was missing a leading parenthesis.]

■ (A) Learn whether two users have the same password with only online computation

■ (B) Learn a specific user's password

■ (C) Change a user's password without detection

☐ (D) Learn every user's password

☐ (E) None of the above

☐ (F) ——

Solution: Mallory only has enough total computation to learn a single user's password, denoted as pwd'. She can now edit a different user's password to be this by computing $\mathsf{H_S}(\mathsf{pwd}' \,\|\, \mathsf{uname})$ and using the signature $\mathsf{Sign}(\mathsf{k}, \mathsf{H_S}(\mathsf{pwd}')))$. Note this is possible because the signature isn't bound to any specific user.

An earlier version of the solutions incorrectly marked (A) as incorrect. However, since signatures are unsalted, an attacker can learn if two users have the same password by comparing signatures (which requires no computation).

Q3.8 (3 points) Describe a DoS attack Mallory can launch against Bob's server if he uses the scheme in Q3.7.

Solution: Basic amplification attack - Mallory makes a bunch of invalid logins which causes Bob to attempt to verify many signatures.

Q3.9 (3 points) Bob decides to add two-factor authentication to the scheme in Q3.7. Does this change your answer to Q3.7?

(A) Yes [✓ (B) No] (C) —— (D) —— (E) —— (F) ——

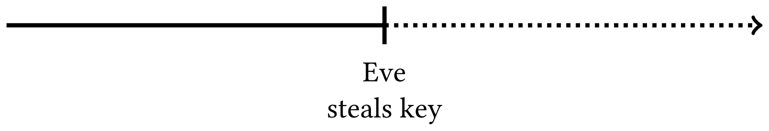
Solution: Two factor authentication prevents an attacker from logging in if they know the password, doesn't help with preventing the attacks mentioned previously.

This is the end of Q3. Proceed to Q4 on your answer sheet.

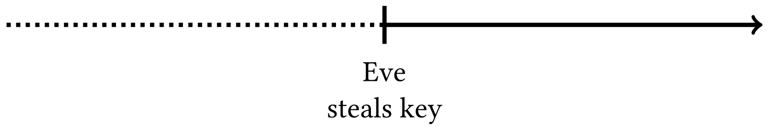
Q4 *Forwards, Backwards, ~~Left, and Right~~* (16 points)

Consider the following properties. The solid part of each timeline denotes the time frame where messages remain confidential, even after Eve, an on-path eavesdropper, steals a key.

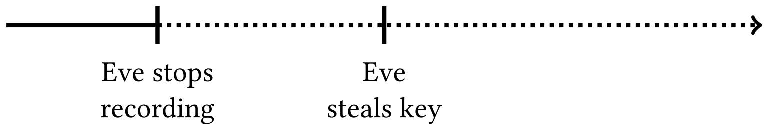
- *Forward secrecy*: If Eve steals a key, past messages remain confidential.



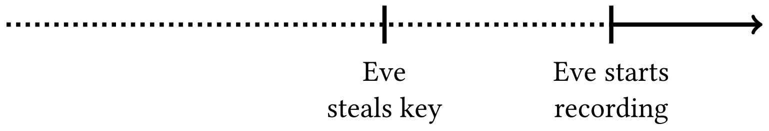
- *Backward secrecy*: If Eve steals a key, future messages remain confidential.



- *Weak forward secrecy*[2]: If Eve stops recording messages, then steals a key, any messages Eve recorded before she stopped recording remain confidential.



- *Weak backward secrecy*[3]: If Eve steals a key, then starts recording messages, any messages Eve record remain confidential.



Consider the following modified symmetric encryption schemes where Alice and Bob change their encryption key for each message they send. For each scheme, determine which of the given properties is ensured. Assume that all keys are 128 bits long, and no party will send more than one message in a row.

Q4.1 (4 points) Alice and Bob increment their shared key k by 1 for each new message, so $k' = k + 1$.

- ☐ (A) Forward secrecy
- ☐ (B) Backward secrecy
- ☐ (C) Weak forward secrecy
- ☐ (D) Weak backward secrecy
- ■ (E) None of the above
- ☐ (F) ——

[2] *Weak forward secrecy* in practice requires that Eve be able to MITM past communication before key compromise, rather than just eavesdropping.

[3] This is a coined term for the purposes of this question.

Solution: Eve can increment and decrement her stolen key in order to attain both past and future keys.

Q4.2 (4 points) Alice and Bob's current shared key is k. For each new message, the sender generates a small, 8-bit random number n and attaches it to the message before encryption. The next message will be encrypted under key $k' = k \oplus \mathsf{PRG}(n)[:128]$, where PRG is a secure PRG.

☐ (G) Forward secrecy

☐ (H) Backward secrecy

☐ (I) Weak forward secrecy

☐ (J) Weak backward secrecy

■ (K) None of the above

☐ (L) ——

Solution: Even though the amount that the key is incremented each time is encrypted, the seed space is small enough for Eve to search through all possible future keys even without access to past or future messages.

Q4.3 (4 points) Alice and Bob's current shared key is k. For each new message, the sender generates a new symmetric key k' and attaches it to the message before encryption. The next message will be encrypted under k'.

■ (A) Forward secrecy

☐ (B) Backward secrecy

■ (C) Weak forward secrecy

■ (D) Weak backward secrecy

☐ (E) None of the above

☐ (F) ——

Solution: If Eve has accesses to all messages, she also has access the key for the next message k', allowing her to decrypt future messages as long as she records every message. She also still has no way of determining what the keys for the previous messages are, since they are randomly generated and have no relation to the given message.

An earlier version of the solutions incorrectly marked A, B, D as the correct answers.

Q4.4 (4 points) For each new message, Alice and Bob conduct Diffie-Hellman key exchange to generate a new symmetric key.

■ (G) Forward secrecy

■ (H) Backward secrecy

■ (I) Weak forward secrecy

■ (J) Weak backward secrecy

☐ (K) None of the above

☐ (L) ——

Solution: An on-path attacker cannot learn the value of the shared key in Diffie-Hellman key exchange. Since a new Diffie-Hellman shared key is generated for every message, even if Eve steals the key for one message, she knows nothing about any messages before or after that message.

This is the end of Q4. Proceed to Q5 on your answer sheet.

Q5 *EvanBotOS* (25 points)

EvanBot is building a new OS and wants to defend against buffer overflow attacks. Bot decides to use cryptography to secure values on the stack.

Assume any cryptography is executed separately and securely by the OS. This means that any cryptographic operations do not count as function calls on the program's stack, and the attacker cannot see the operations being executed. Also, unless otherwise stated, **any MACs or hashes generated are stored separately in the OS, not on the stack**.

Assume stack canaries are four random bytes (no null byte). Assume the OS has a secret key k that is unknown to any attacker.

For each part, mark which scheme is more secure (would defend against more buffer overflow attacks), or if both schemes would defend against the same set of attacks.

[*Clarification during exam*: For each scheme, unless otherwise specified all memory safety defenses are disabled.]

Q5.1 (3 points) Scheme A: When a function is called, push a random stack canary to the stack. Also, generate a MAC on the canary value using k. Before the function returns, in addition to checking that the canary is the same, also verify the canary with the MAC.
Scheme B: No cryptography, stack canaries are enabled, WˆX and ASLR are disabled.

○ (A) Scheme A [✓ (C) The same] ○ (E) ——

○ (B) Scheme B ○ (D) —— ○ (F) ——

Solution: Any exploit on Scheme B would need to have the canary value be unchanged before the function returns (either by overwriting the canary with itself, writing around the canary, or brute-forcing the canary). If the canary value is unchanged, using a MAC on the canary won't detect an exploit that changes other parts of the stack.
A bug in this question was discovered during the exam. For Scheme B, in practice, most compilers generate one stack canary per program, and the canary value is the same for every function. (We did not explicitly cover this in lecture this semester.) However, the wording of this question suggests that in Scheme A, the stack canaries are different for every function in one program. Under this interpretation, Scheme A would be better, since it does not reuse stack canaries. For this reason, we accepted Scheme A as an alternate valid answer.

Q5.2 (3 points) Scheme A: When a function is called, encrypt a randomly-generated stack canary using k. Push the encrypted canary onto the stack. Before the function returns, decrypt the stack canary and verify that it is unchanged.
Scheme B: No cryptography, stack canaries are enabled, WˆX and ASLR are disabled.

○ (G) Scheme A [✓ (I) The same] ○ (K) ——
○ (H) Scheme B ○ (J) —— ○ (L) ——

Solution: Both schemes are powerless against exploits that don't involve the canary or write around the canary. For exploits involving the canary, the encryption step doesn't add any extra security - from the attacker's perspective, the canary is still four random bytes that need to be left unchanged (by overwriting them with itself or brute-forcing).
This subpart has the same bug as the subpart above. We accepted Scheme A as an alternate valid answer.

Q5.3 (3 points) Scheme A: When a program is first started, generate a signature on every page of the memory space using k. If the program tries to execute any instructions in memory, check that the page where the instruction is stored is correctly signed.
Scheme B: No cryptography, WˆX is enabled, stack canaries and ASLR are disabled.

○ (A) Scheme A [✓ (C) The same] ○ (E) ——
○ (B) Scheme B ○ (D) —— ○ (F) ——

Solution: Scheme A prevents any data written into memory from being executed (because it won't be signed). This is equivalent to the functionality of the WˆX bit.

Q5.4 (3 points) Scheme A: When a function is called, using a cryptographic hash H, hash the RIP, *and push the value of the hash onto the stack*. Before the function returns, verify that the RIP still hashes to the same value.
Scheme B: When a function is called, generate a MAC on the RIP using k, *and push the value of the MAC onto the stack*. Before the function returns, verify the RIP with the MAC.
Assume that the hash and the MAC are the same length.

○ (G) Scheme A ○ (I) The same ○ (K) ——
[✓ (H) Scheme B] ○ (J) —— ○ (L) ——

Solution: Scheme A doesn't provide any extra protection because an attacker can hash the malicious RIP and overwrite the original hash with the hash of the malicious RIP. In Scheme B, the attacker cannot forge a MAC for the RIP because the attacker doesn't have the value of k.

Q5.5 (5 points) Consider Scheme A from the previous part. Briefly explain how you might create an exploit for Scheme A that overwrites the RIP. Assume you can debug only the vulnerable program with GDB, and you cannot access the OS-level cryptography operations.

○ (A) —— ○ (B) —— ○ (C) —— ○ (D) —— ○ (E) —— ○ (F) ——

Solution: As above, just hash the malicious RIP and overwrite the original hash with the hash of the malicious RIP.

Q5.6 (3 points) Scheme A: When a function is called, encrypt the RIP with a one-time pad, where the pad is a static value stored in the OS. (The pad value does not change when you rerun the program.) Before the function returns, decrypt the RIP and jump to that location.
Scheme B: No cryptography, stack canaries are enabled, W^X and ASLR are disabled.

○ (G) Scheme A
[✓ (H) Scheme B]
○ (I) The same
○ (J) ——
○ (K) ——
○ (L) ——

Solution: OTP with key reuse is insecure, so it's equivalent to not using any defenses at all.

Q5.7 (5 points) Consider Scheme A from the previous part. In 2-3 sentences, explain how you might create an exploit for Scheme A that overwrites the RIP. Assume you can debug only the vulnerable program with GDB, and you cannot access the OS-level cryptography operations.

○ (A) —— ○ (B) —— ○ (C) —— ○ (D) —— ○ (E) —— ○ (F) ——

Solution: In GDB, overwrite the RIP with 0x00000000. This will cause the program to try and jump to PAD $\oplus$ 0x00000000 = PAD. Now that you know the pad, just XOR the desired address with the pad when performing the exploit.
Note that solutions that don't overwrite the RIP with a known value will not work, since the RIP is encrypted with the OTP, and even if you ran the program twice, you would only see the same encrypted RIP twice.
An alternate solution is to disassemble the entire set of instructions, look for a call instruction that calls the currently executing function, and then deduce the value of RIP based on where the call instruction is located. But this would take a lot of trial-and-error, especially if the currently executing function is called several times.

This is the end of Q5. Proceed to Q6 on your answer sheet.

Q6 *DNS over TCP* (20 points)

Standard DNS uses UDP to send all queries and responses. Consider a modified DNS that instead uses TCP for all queries and responses.

Q6.1 (3 points) Which of the following does DNS over TCP guarantee against a man-in-the-middle attacker? Select all that apply.

☐ (A) Confidentiality ☐ (C) Authenticity ☐ (E) ——

☐ (B) Integrity [✓ (D) None of the above] ☐ (F) ——

Solution: TCP has no cryptographic guarantees, so a MITM attacker can read and modify any message.

Q6.2 (3 points) Compared to standard DNS, does DNS over TCP defend against more attacks, fewer attacks, or the same amount of attacks against an on-path attacker?

○ (G) More attacks ○ (I) Fewer attacks ○ (K) ——

[✓ (H) Same amount of attacks] ○ (J) —— ○ (L) ——

Solution: An on-path attacker can see all relevant header fields in TCP and UDP, so they only need to win the race against the legitimate response in both standard DNS and DNS over TCP.

Q6.3 (5 points) What fields does an off-path attacker ***not know*** and need to ***guess*** correctly to spoof a response in DNS over TCP? Assume source port randomization is enabled. Select all that apply.

[✓ (A) TCP sequence numbers] [✓ (C) Recursive resolver port] ☐ (E) DNS NS records

☐ (B) Name server port ☐ (D) DNS A records ☐ (F) None of the above

Solution: To spoof a TCP packet, the off-path attacker needs to guess the TCP sequence numbers and the randomized resolver port (source port). The name server port (destination port) is public and well-known. The DNS records can be anything the attacker wants, so there is nothing to guess there.

Q6.4 (3 points) Is the Kaminsky attack possible on DNS over TCP? Assume source port randomization is disabled.

○ (G) Yes, because the attacker only needs to guess the DNS Query ID

[✓ (H) Yes, but we consider it infeasible for modern attackers]

○ (I) No, because the attacker cannot force the victim to generate a lot of DNS over TCP requests

○ (J) No, because TCP has integrity guarantees

○ (K) ——

○ (L) ——

○ (M) ——

Solution: The attacker would have to guess at least 32 bits of sequence numbers, which is the same defense as source port randomization in standard DNS.

Q6.5 (3 points) Recall the DoS amplification attack using standard DNS packets. An off-path attacker spoofs many DNS queries with the victim's IP, and the victim is overwhelmed with DNS responses.

Does this attack still work on DNS over TCP?

○ (A) Yes, the attack causes the victim to consume more bandwidth than the standard DNS attack

○ (B) Yes, the attack causes the victim to consume less bandwidth than the standard DNS attack

○ (C) No, because the DNS responses no longer provide enough amplification

[✓ (D) No, because the attacker cannot force the server to send DNS responses to the victim]

○ (E) ——

○ (F) ——

Solution: To force the victim to receive a DNS response, the attacker would need to initiate a TCP connection that looks like it's from the victim. However, an off-path attacker cannot do this, since they cannot see the SYN-ACK response sent to the victim.

Q6.6 (3 points) What type of off-path DoS attack from lecture is DNS over TCP vulnerable to, but standard DNS not vulnerable to? Answer in five words or fewer.

Solution: TCP SYN Flooding

Q7 *I (T)C(P) You* (26 points)

EvanBot builds a new course feature that sends announcements to students over TCP. To receive announcements, a student initiates a TCP connection with the server. The server sends the announcements and terminates the connection.

Q7.1 (3 points) Assuming that no adversaries are present, which of the following does communication over a TCP connection guarantee? Select all that apply.

■ (A) That both the server and client can detect if a particular announcement needs to be resent

■ (B) That different announcements are delivered in the same order they were sent in

☐ (C) That announcements are delivered using the most efficient path through the internet

☐ (D) None of the above

☐ (E) ——

☐ (F) ——

Solution: TCP guarantees that messages will be retransmitted until they are successfully delivered, and that messages will be delivered in the correct order. TCP makes no guarantees about what path a packet takes through the Internet.

Q7.2 (3 points) When only an on-path adversary is present, which of the following does communication over a TCP connection guarantee? Select all that apply.

☐ (G) That both the server and client can detect if a particular announcement needs to be resent

☐ (H) That different announcements are delivered in the same order they were sent in

☐ (I) That announcements are delivered using the most efficient path through the internet

■ (J) None of the above

☐ (K) ——

☐ (L) ——

Solution: An on-path attacker has access to the TCP sequence numbers, so they can inject arbitrary messages. Since the attacker can interfere with all messages, TCP no longer has any guarantees about message delivery. TCP still makes no guarantees about what path a packet takes through the Internet.

Q7.3 (3 points) Suppose that EvanBot instead sends announcements over UDP. Assuming that no adversaries are present, which of the following might happen? Select all that apply.

■ (A) Students might not receive some announcements

■ (B) Students might receive the announcements more quickly

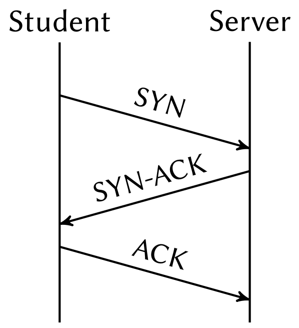
■ (C) The server might not detect some errors which it would have had it been using TCP

☐ (D) None of the above

☐ (E) ——

Solution: UDP no longer guarantees delivery, so some announcements might not be delivered. However, UDP does not require a handshake at the beginning, so announcements can be delivered more quickly. UDP has no guarantees about what order announcements arrive in, so the server will no longer detect if packets arrive out of order.

EvanBot realizes that the server is sending messages to the student, but the student only responds with ACKs and never sends any messages after the initial handshake. They design a *Half TCP* protocol which provides TCP's properties for communications from the server to the student, but not for communications from the student to the server. This is accomplished using a modified version of the standard three step handshake pictured below.



Q7.4 (5 points) Some sequence numbers are no longer necessary in *Half TCP*. Which fields **do not** need to be transmitted? Select all that apply.

■ (G) The sequence number in the SYN packet

☐ (H) The sequence number in the SYN-ACK packet

■ (I) The ACK number in the SYN-ACK packet

■ (J) The sequence number in the ACK packet

☐ (K) The ACK number in the ACK packet

☐ (L) None of the above

Solution: The key insight here is that because the student isn't sending messages to the server, the student's sequence numbers are no longer necessary. The SYN and ACK packets are sent from the student to the server, so their sequence numbers are no longer necessary. The SYN-ACK packet is sent from the server to the student, so its ACK number is no longer necessary.

An earlier version of the solutions incorrectly marked H, K as the set of correct answers. When revising the exam, we changed the question to be "which fields **do not** need to be transmitted," which caused the set of correct answers to be inverted.

Q7.5 (3 points) Which of these are consequences of moving from TCP to *Half TCP* for this application? Select all that apply.

☐ (A) The student will no longer receive announcements in the correct order

■ (B) The server will not have to keep track of as much state

■ (C) The student will not have to keep track of as much state

☐ (D) None of the above

☐ (E) ——

☐ (F) ——

Solution: Announcements are sent from the server to the student. We are still using sequence numbers in this direction, so the announcements are still received in the correct order. Because the server and student each only need to keep track of one sequence number instead of two, they both do not need to keep track of as much state.

The 161 staff likes security and decides to use TLS over *Half TCP*. Assume that the staff server has a valid certificate for their public key.

For each different adversary below, select all attacks which become ***easier*** when running TLS over *Half TCP* compared to normal TCP.

Q7.6 (3 points) Off-path adversary

■ (G) RST Injection Attack

☐ (H) Interfere with a TLS handshake to learn the master key

☐ (I) Replay an encrypted command from a previous TLS connection

☐ (J) None of the above

☐ (K) ——

☐ (L) ——

Q7.7 (3 points) On-path adversary

☐ (A) RST Injection Attack

☐ (B) Interfere with a TLS handshake to learn the master key

☐ (C) Replay an encrypted command from a previous TLS connection

■ (D) None of the above

☐ (E) ——

☐ (F) ——

Q7.8 (3 points) Man-in-the-middle adversary

☐ (G) RST Injection Attack

☐ (H) Interfere with a TLS handshake to learn the master key

☐ (I) Replay an encrypted command from a previous TLS connection

■ (J) None of the above

☐ (K) ——

☐ (L) ——

Solution: The key insight here is that attacks on the TLS protocol are not made any easier by using half-TCP, because the cryptographic messages sent between the student and the server are unchanged. The only attack that becomes easier is the RST injection attack for an off-path attacker, since the attacker doesn't need to guess sequence numbers when injecting a RST packet from the student to the server. On-path and MITM attackers can see all sequence numbers, so RST injection is not any easier for them.

This is the end of Q7. Proceed to Q8 on your answer sheet.

Q8 *Election Security* (23 points)

The 2020 elections are coming up, and the United States Government has tasked you with securing the nation's voting machines!

Assume election headquarters are in a top-secret, undisclosed site. All incoming network requests pass through a network-based intrusion detection system (NIDS), as well as a firewall. Outside users can only access the server with HTTPS.

Q8.1 (3 points) Which of these attacks are **always** preventable in this setup? Assume the attacker is on-path. Select all that apply.

- ☐ (A) RST Injection Attack
- ☐ (B) SQL Injection Attack
- ☐ (C) Reflected XSS Attack
- [✓ (D) None of the Above]
- ☐ (E) —
- ☐ (F) —

Q8.2 (3 points) Which of these attacks are **always** preventable in this setup? Assume the attacker is on-path. Select all that apply.

- [✓ (G) SYN Flooding Attack]
- ☐ (H) DNS Spoofing Attack
- ☐ (I) DDoS Attack
- ☐ (J) None of the Above
- ☐ (K) —
- ☐ (L) —

Solution:

- RST Injection Attack - HTTPS doesn't prevent RST Injection attacks, so they're still a potential vulnerability
- SQL Injection Attack - these attacks are generally application-layer (so transport-layer security and firewalls don't protect against them)
- Reflected XSS Attack - same reasoning as above. Additionally, even if NIDS were capable of detecting these over HTTP, it wouldn't be able to see any payloads under HTTPS.
- SYN Flooding Attack - these attacks are preventable using SYN Cookies!
- DNS Spoofing Attack - none of the defenses prevent DNS Spoofing
- DDoS Attack - not much a NIDS can do here, unfortunately

Q8.3 (3 points) An attacker injects malicious code on a server inside the election headquarters that changes all submitted votes to one candidate. Which detection system is best suited to defend against this attacker?

- [✓ (A) HIDS]
- ○ (B) NIDS
- ○ (C) Firewall
- ○ (D) —
- ○ (E) —
- ○ (F) —

Solution: Only a host-based system would be able to detect and/or prevent this attack from happening!

Q8.4 (3 points) An attacker realizes that the ballot boxes are running a vulnerable version of Linux, and uses a previously-known buffer overflow exploit. Which detection method is best suited to defend against this attacker?

○ (G) Anomaly-Based Detection

[✓ (H) Signature-Based Detection]

○ (I) Specification-Based Detection

○ (J) Behavioral-Based Detection

○ (K) ——

○ (L) ——

Solution: Signature-based detection approaches are primarily responsible for catching known attacks!

Q8.5 (5 points) Ben, a computer scientist at the top-secret site, has a HIDS installed on his work laptop. He decides to sign into his personal email account, claiming that HTTPS will protect the government from seeing his emails. Is he correct? Justify your answer in 1–2 sentences.

○ (A) Yes

[✓ (B) No]

○ (C) ——

○ (D) ——

○ (E) ——

Solution: Host-based intrusion detection systems are capable of reading data inbound/outbound HTTPS connections, so Ben's use of HTTPS doesn't really help him here.

We also accepted yes as an answer if it was justified by claiming he could use an email client that the HIDS didn't have access to

Q8.6 (3 points) You're discovered that an attacker has managed to connect to a service running inside our network from IP Address 5.6.7.8 and is in the process of performing a DoS attack! Write a stateful firewall rule to block all traffic originating from the attacker. Our service is running on IP address 1.2.3.4 (port 443).

Solution: drop * 5.6.7.8 :*/ext -> 1.2.3.4 :443/int

Q8.7 (3 points) You've received a tip that attackers have devised a plan to spoof ballot submissions. Here's the information that your source provides:

- 20 out of every 100 submissions are malicious.
- The cost to investigate an incorrectly flagged submission is $5.

- The cost of letting a spoofed submission through is \$50.

You're offered two different intrusion detection systems. System A offers a false positive rate of 10% and a false negative rate of 25%. System B offers a false positive rate of 50% and a false negative rate of 5%. Which do you choose?

○ (A) System A

[✓ (B) System B]

○ (C) Not enough information

○ (D) Either system

○ (E) —

○ (F) —

Solution: The expected cost per 100 submissions:

- System A:
$$(0.10) * (80) * (5) + (0.25) * (20) * (50) = 290$$

- System B:
$$(0.50) * (80) * (5) + (0.05) * (20) * (50) = 250$$

So System B is better

This is the end of Q8. Proceed to Q9 on your answer sheet.

Q9 *Cookie Debugger* (37 points)

EvanBot is adding a feature on the CS161 course website that lets students log in and view their grades. However, Bot forgot to remove a debugging feature–if anyone visits `cs161.org/debug`, the webpage will display all the cookies sent to the server.

Assume the `cs161.org/debug` page does not have any other functionality. Assume anyone can create an account on the website. Each subpart is independent.

Q9.1 (3 points) Which of the following URLs have the same origin as `http://cs161.org/debug` according to the same-origin policy?

■ (A) `http://cs161.org/`

☐ (B) `http://cs161.org:8081/debug`

☐ (C) `https://cs161.org/debug`

☐ (D) None of the above

☐ (E) ——

☐ (F) ——

Solution: Two sites must have identical protocols, hostnames, and ports in order for them to be qualified as having the same origin (under the SOP). In this case, the two options that do not work are the one with **Port 8081**, and the one with protocol `https://`. Note: SOP is not affected by the URL Path.

Q9.2 (5 points) Which of the following cookies would be displayed when visiting `https://cs161.org/debug`? Assume the client's origin is `https://cs161.org`.

■ (G) Domain = `cs161.org`, Path = `/`, `Secure`

■ (H) Domain = `cs161.org`, Path = `/`, `HttpOnly`

☐ (I) Domain = `debug.cs161.org`, Path = `/`, `Secure`, `HttpOnly`

■ (J) Domain = `cs161.org`, Path = `/debug`

■ (K) Domain = `cs161.org`, Path = `/`, `SameSite=strict`

☐ (L) None of the above

Solution: The `HttpOnly` attribute is irrelevant here, because we're not concerned with modifying the cookie in JavaScript.

The `Secure` attribute is also irrelevant here, since we are using HTTPS and the cookie will be sent regardless of whether the `Secure` attribute is set.

The domains and paths are valid in all options, so all cookies will be displayed when sent.

Q9.3 (3 points) Suppose you set a cookie `test=<script>alert("This exam is hard!")</script>` with valid attributes, and load `https://cs161.org/debug`. A pop-up that says `This exam is hard!` appears in your browser. Have you successfully found a server vulnerability?

[*Clarification during exam*: The pop-up had a typo in it.]

[✓ (A) Yes, you found an XSS vulnerability]

(B) Yes, you found a CSRF vulnerability

(C) No, because you have not changed any state on the server side

(D) No, because the JavaScript does not run with the origin of `cs161.org`

(E) —

(F) —

Q9.4 (5 points) Consider a modification to the course website. Before rendering any page, the server searches for every pair of `<script>` and `</script>` tags and removes the tags *and everything between the tags.*

Can you still cause JavaScript to run in your browser using `<script>` tags? If yes, provide a cookie name and value (written as `name=value`) that would cause `alert(1)` to run. If no, briefly explain why.

[✓ (G) Yes] (H) No (I) — (J) — (K) — (L) —

Solution: Yes. Consider the cookie `<scr<script></script>ipt>alert(1)</script>`. After removing the `<script></script>` tags and everything in between, you're left with `<script>alert(1)</script>`.

Q9.5 (5 points) Consider a modification to the course website. Before rendering any page, the server renders the cookie name in an isolated environment and ensures that no scripts are run, and then does the same for the cookie value.

Assume that the website displays the cookie name and value with no added text in between. Can you still cause JavaScript to run in your browser using `<script>` tags? If yes, provide a cookie name and value (written as `name=value`) that would cause `alert(1)` to run. If no, briefly explain why.

[✓ (A) Yes] (B) No (C) — (D) — (E) — (F) —

Solution: Yes. Set the cookie name to `<script>alert(1)` and the cookie value to `</script>`. Then neither part of the cookie runs a script in a sandbox, but together they cause the script to run.

Q9.6 (3 points) Is it possible to create a link to `cs161.org/debug` that will cause another user to run malicious JavaScript when they click on the link?

(G) Yes, because you can place JavaScript in the HTTP GET parameters

○ (H) Yes, because you can place JavaScript in the HTTP POST body

[✓ (I) No, because there is nowhere to place the JavaScript]

○ (J) No, because the server is secure against this attack

○ (K) ——

○ (L) ——

Solution: The `cs161.org/debug` webpage only displays cookies, not any HTTP GET parameters or HTTP POST body. Cookies cannot be attached in a malicious URL.

Q9.7 (5 points) Suppose a victim visits the attacker-controlled `evil.cs161.org`. Write a JavaScript snippet that would cause the victim to run `alert(1)` in their browser with the origin of `cs161.org`. If you don't know the exact Javascript syntax, pseudo-code is acceptable.

Hint: `window.location = "google.com";` in JavaScript causes the user to load `google.com`.

Solution:

```
<script>
document.cookie="test=<script>alert(1)</script>;domain=cs161.org;path=/";
window.location = "cs161.org/debug";
</script>
```

The first part of the script sets a cookie that would cause `alert(1)` to run, with the appropriate domain and path. The second part of the script causes the user to load `cs161.org/debug` with the malicious cookie.

Q9.8 (5 points) Which of the following malicious pages would be able to run your Javascript exploit against the user?

[✓ (G) `http://very.evil.cs161.org/`]

[✓ (H) `http://very-evil.cs161.org/`]

☐ (I) `http://evil-cs161.org/`

[✓ (J) `http://cs161.org/evil`]

☐ (K) `http://evil.com/`

☐ (L) None of the above

Solution: `very.evil.cs161.org`, `very-evil.cs161.org`, and `cs161.org` all contain the `cs161.org` domain suffix, so they are able to set the XSS cookie and execute the attack. Note that the path is irrelevant.

Q9.9 (3 points) Consider a modification to the course website. The `cs161.org/debug` page only displays cookies if the request contains a valid session token. Does your Javascript exploit still work?

○ (A) Yes, with no modifications

[✓ (B) Yes, with minor modifications (changing 1-2 lines of code)]

(C) No

(D) ——

(E) ——

(F) ——

Solution: The attacker could create an account, receive a session token, and set a cookie in the victim's browser with that session token. This will cause the victim's request to look like it came from the attacker, but the JavaScript will still run in the victim's browser.

This is the end of Q9. Proceed to Q10 on your answer sheet.

Q10 *Bitcoin* (12 points)

Assume a simplified Bitcoin model, where each block contains the following fields:

- `minerID`: The public key of the node who mined this block. Recall that the person who mined a block is given a mining reward in Bitcoin. Assume that a miner can redeem this award by simply referencing the block ie. the initial award is *not* stored as a transaction.
- `prevHash`: The hash of the previous block
- `transactions`: The list of transactions. Recall each transaction contains references to its origin transactions, a list of recipients, and is signed using the private key of the coins' owner.
- `nonce`: A value such that the hash of the current block contains the correct number of zeros

Assume that the hash of a block is computed as:

`Hash(minerID || prevHash || transactions || nonce)`

Bob wants to save on computing power by omitting certain fields in a block from being part of the hash. For each modified block hashing scheme below, select all the things an adversary with a single standard CPU can do.

Assume that if the adversary can come up with a modified blockchain of the same length, the rest of the network will accept it. Furthermore, assume the adversary has not made any transactions thus far. **Any option that could result in an invalid state should not be selected.**

Q10.1 (4 points) Each block hash is computed as `Hash(prevHash || transactions || nonce)`

■ (A) Modify a block to gain Bitcoin

☐ (B) Given some amount of pre-computation, can consistently win proof of work

☐ (C) Modify some transaction amounts

☐ (D) Can remove any transaction in an arbitrary block by *only* modifying that block

☐ (E) None of the above

☐ (F) ——

Solution: An adversary can change the `minerID` of some past blocks to give themselves the mining reward. Note that this mining reward can't be used in a subsequent transaction or else we would reach an invalid state, but, at the very least, the most recently added block will always have a mining reward that hasn't been spent yet.

Q10.2 (4 points) Each block hash is computed as `Hash(minerID || transactions || nonce)`

■ (G) Modify a block to gain Bitcoin

■ (H) Given some amount of pre-computation, can consistently win proof of work

☐ (I) Modify some transaction amounts

☐ (J) Can remove any transaction in an arbitrary block by *only* modifying that block

☐ (K) None of the above

☐ (L) ——

Solution: Like before, an adversary can change any `minerID`s that haven't been spent yet since blocks no longer have a requirement on the past chain.

They can also precompute a valid nonce for a block they want to add, since the hash is independent of the chain.

Since the blocks aren't directly dependent on eachother anymore, the adversary can change any individual block. *However*, they can't remove a transaction if a future transactions makes use of it (this would be an invalid state).

They cannot modify a transaction amount because each transaction is signed.

Q10.3 (4 points) Each block hash is computed as `Hash(minerID || prevHash || nonce)`

☐ (A) Modify a block to gain Bitcoin

☐ (B) Given some amount of pre-computation, can consistently win proof of work

☐ (C) Modify some transaction amounts

☐ (D) Can remove any transaction in an arbitrary block by *only* modifying that block

[✓ (E) None of the above]

☐ (F) ——

Solution: We can't modify `minerID`s anymore since the blockchain has dependence on them.

We can't consistently win PoW via pre-computation since the blocks form a blockchain.

We can't remove any transaction in an arbitrary block as this might cause an invalid state and we can't modify transaction amounts because of signatures

This is the end of Q10. Proceed to Q11 on your answer sheet.