# Popa & Wagner Spring 2020 — CS 161 Computer Security — Midterm 2

**This exam was generated for `foo@bar.com`.**

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 *zero* or more choices on Gradescope.

■ You can select

■ multiple squares

For questions with a **large box**, you need to provide justification in the text box on Gradescope.

You have 80 minutes. There are 10 questions of varying credit (200 points total).

The exam is open book. You can use any resources on the Internet, including course notes, as long as you are working alone.

We will not be answering any clarifications about the exam. If there are any glaring problems with wording, we will consider dropping the question from the exam after solutions/grades are released.

## Q1 *MANDATORY – Honor Code* **(3 points)**

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

**This is the end of Q1. Proceed to Q2 on your Gradescope answer sheet.**

## Q2 *True/False* **(48 points)**

Each true/false is worth 2 points unless otherwise specified.

Q2.1 TRUE or FALSE: If Bob uses the authenticate-then-encrypt paradigm, the integrity of his plaintext is guaranteed.

● TRUE ○ FALSE

**Solution:** True. Authenticate-then-encrypt guarantees integrity for the plaintext, just not the ciphertext.

Q2.2 TRUE or FALSE: A hash function must be collision-resistant to be considered safe for password hashing.

○ TRUE ● FALSE

**Solution:** False. Only the one-way property is needed. Collisions are okay as long as one cannot find a preimage for a given function value.

Q2.3 TRUE or FALSE: Alice wants to use authenticated encryption to send a message to Bob. She should use Enc(M), MAC(H(Enc(M))) over Enc(M), MAC(H(M)).

● TRUE ○ FALSE

**Solution:** True. The former scheme provides confidentiality, whereas the second scheme may not, depending on what MAC algorithm is used. For instance, MAC(H(M)) may be a deterministic function of M, so an eavesdropper might be able to identify when the same message is sent twice. Also the second scheme may be vulnerable to padding oracle attacks if it is not implemented very carefully, though this is beyond the scope of this class.

Q2.4 Suppose we increase the entropy of the DNS ID field to 128 bits. It is infeasible for an on-path adversary to spoof a DNS answer.

○ TRUE ● FALSE

**Solution:** False. The adversary is on-path so they can see the QID in plaintext.

Q2.5 TRUE or FALSE: By default, in a TLS connection, both the server and client are authenticated to each other.

○ TRUE ● FALSE

**Solution:** False. TLS only authenticates the server by default.

Q2.6 TRUE or FALSE: If weak passwords are salted and hashed before being stored, then the attacker cannot easily learn the plaintext values of the weak passwords.

○ TRUE ● FALSE

**Solution:** False. The salted password is still susceptible to a dictionary attack.

Q2.7 TRUE or FALSE: A DNS lookup for `en.wikipedia.org` will always force the recursive resolver to send at least 3 DNS queries.

○ TRUE ● FALSE

**Solution:** False. Answers could be cached.

Q2.8 TRUE or FALSE: If the server's random number $a$ in Diffie-Hellman TLS is the same in every handshake, Diffie-Hellman TLS no longer has forward secrecy. Assume the value $a$ is stored on the server along with its secret key.

● TRUE ○ FALSE

**Solution:** True. An attacker who steals $a$ will be able to reconstruct the PS and decrypt past recorded messages by computing $(g^b)^a \bmod p$.

Q2.9 TRUE or FALSE: If Bob is an on-path attacker who can guarantee that his spoofed response arrives before the legitimate response, Bob only needs the victim to make one request for a nonexistent domain in order to successfully execute a Kaminsky attack with 100% probability.

● TRUE ○ FALSE

**Solution:** True. Bob doesn't need to guess any random fields or try several times to race the legitimate response.

Q2.10 TRUE or FALSE: Randomizing the client port helps defend TCP against on-path attackers.

○ TRUE ● FALSE

**Solution:** False. The on-path attacker can see the port values.

Q2.11 TRUE or FALSE: TLS provides end-to-end security, so it is secure even if the server has a buffer overflow vulnerability.

○ TRUE ● FALSE

**Solution:** False. If an attacker exploits the buffer overflow vulnerability to gain control of the server, TLS doesn't stop you from talking to the compromised server.

Q2.12 TRUE or FALSE: Suppose we modified TCP so that the sequence number increases by 2 for every byte sent, but the initial sequence numbers are still randomly chosen. This modified protocol has the same security guarantees as standard TCP.

● TRUE ○ FALSE

**Solution:** True. Incrementing the sequence number differently doesn't make it any easier for an off-path attacker to guess it, and if you're on-path or MITM, you can see everything anyway.

Q2.13 TRUE or FALSE: If IP spoofing is eliminated from the Internet (all attackers must send messages from their real IP), then an on-path attacker is no longer more powerful than an off-path attacker.

○ TRUE ● FALSE

**Solution:** False. On-path attackers can still read messages, while off-path attackers can't.

Q2.14 TRUE or FALSE: Consider a modified version of DHCP, where in the server offer step, the server signs its message and sends its public key along with the signed message. This version of DHCP is secure against the DHCP spoofing attack.

○ TRUE ● FALSE

**Solution:** False. The client has no way to verify the public key. An attacker could easily send their own malicious public key and use that to sign a spoofed response.

Q2.15 TRUE or FALSE: TCP is secure against a DoS attack by a man-in-the-middle (MITM) because TCP guarantees delivery and will re-send messages until they are delivered.

○ TRUE ● FALSE

**Solution:** False. The MITM could just keep dropping packets so that messages never arrive. Also, the MITM can inject a RST packet, which ends the connection.

Q2.16 TRUE or FALSE: RSA-TLS is still secure if we use publically known lottery numbers as the value of the premaster secret (PS).

○ TRUE ● FALSE

**Solution:** False. An on-path or MITM attacker would know $R_b$ and $R_s$ (sent in plaintext) as well as PS, which would allow them to generate the symmetric keys and decrypt everything.

Q2.17 TRUE or FALSE: Under the SOP, it is possible for two webpages with different origins to communicate through narrowly defined APIs.

● TRUE ○ FALSE

**Solution:** True. This is the postMessage API.

Q2.18 True or False: Under the SOP, the webpage at `https://example.com/randompic.html` cannot fetch the image at `https://cute-cats.com/cutest.jpg` because they have different origins.

○ True ● False

**Solution:** False. A page can fetch images and content regardless of origin; the SOP only prevents it from determining detailed properties of cross-origin content.

Q2.19 True or False: Suppose the webpage at `https://example.com/index.html` contains a child frame that loads `https://another-example.com/index.html`. Under the SOP, the parent frame can read and modify the properties of the child frame.

○ True ● False

**Solution:** False. Because `example.com` and `another-example.com` have different domains, they are considered different origins and are therefore unable to access each other directly.

Q2.20 (2 points) True or False: Suppose the webpage at `https://example.com/index.html` contains a child frame that loads `https://example.com/views.html`. Under the SOP, the child frame can read and modify the properties of the parent frame.

● True ○ False

**Solution:** True. Pages are allowed to directly access child frames from the same origin, and vice versa.

Q2.21 True or False: Suppose the webpage at `https://example.com/index.html` loads and runs an external script from `https://sample.com/script.js`. Under the SOP, the script runs with the same origin as `https://sample.com/script.js`.

○ True ● False

**Solution:** False. External scripts run with the origin of the page that fetched them (in this case, `https://example.com`).

Q2.22 True or False: Mallory convinces Alice to connect to her private Wi-Fi network. Webpages that Alice visits while on this network may no longer be subject to the SOP.

○ True ● False

**Solution:** False. The SOP is unrelated to the network.

Q2.23 TRUE or FALSE: Mallory convinces Alice to try out her custom browser, FireFaux. Webpages Alice visits using this browser may no longer be subject to the SOP.

● TRUE ○ FALSE

**Solution:** True. The SOP is enforced by the browser; if the browser is compromised, there is no guarantee that webpages will play by the rules.

Q2.24 TRUE or FALSE: Consider a modified version of Diffie-Hellman TLS where the server does not include the signature when sending $g^a \bmod p$. This version of TLS does not provide confidentiality against a MITM.

● TRUE ○ FALSE

**Solution:** Without the signature, we lose protection against a Man-in-the-Middle since we are now effectively using standard Diffe-Hellman key exchange. An adversary can now pretend to be the client to the server and pretend to be the server to the client, and therefore we lose all security guarantees because we now trust the Man-in-the-Middle.

Q2.25 (0 points) TRUE or FALSE: EvanBot is a bot.

● TRUE ○ FALSE

**Solution:** True. How dare you doubt our trusty AI.

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

## Q3 (18 points)

For each public-key infrastructure (PKI) scheme, mark whether it provides the same trust guarantees as the standard PKI from lecture for all certificates, some certificates, or no certificates at all. Assume that everyone has the root certificate hardcoded into their machines.

Q3.1 (3 points) Each server can only sign the public keys of its grandchildren (two descendants below the current level). For example, the root server can sign the public key of `berkeley.edu` but not `.edu`, and the `.edu` server can sign the public key of `eecs.berkeley.edu` but not `berkeley.edu`.

◯ (A) All certificates ◯ (C) No certificates ◯ (E) ——
● (B) Some certificates ◯ (D) —— ◯ (F) ——

> **Solution:** This works for any certificates located an even number of levels below the root. However, there is no way to create a path of trust from the root to a certificate located an odd number of levels below the root, such as `eecs.berkeley.edu`.

Q3.2 (3 points) As in the previous part, each server can only sign the public keys of its grandchildren. However, the root is additionally allowed to sign the public key of its direct children. For example, the root server can sign the public key of `.edu` and `berkeley.edu`. The `.edu` server can sign the public key of `eecs.berkeley.edu` but not `berkeley.edu`.

● (G) All certificates ◯ (I) No certificates ◯ (K) ——
◯ (H) Some certificates ◯ (J) —— ◯ (L) ——

> **Solution:** Skipping two levels at a time, all certificates must have a path of trust that ends at either root or a server one level below root. Since the root is allowed to sign public keys of servers one level below it, this scheme now works for all certificates.

Q3.3 (3 points) Same setup as the previous part, but an attacker has compromised a server one level below the root (e.g. `.edu`).

◯ (A) All certificates ◯ (C) No certificates ◯ (E) ——
● (B) Some certificates ◯ (D) —— ◯ (F) ——

> **Solution:** Any certificate whose path to the root doesn't go through the compromised server (e.g. `google.com`) is unaffected, but a certificate whose path goes through the compromised server (e.g. `berkeley.edu`) cannot be trusted.

Q3.4 (3 points) The root handles all requests and sends the requested public key and a certificate directly through a TLS connection.

● (G) All certificates ○ (I) No certificates ○ (K) ——
○ (H) Some certificates ○ (J) —— ○ (L) ——

**Solution:** TLS provides end-to-end integrity.

Q3.5 (3 points) Instead of signing, use a cryptographic hash to create a certificate. For example, the root server signs the public key of `.edu` by hashing it.

○ (A) All certificates ● (C) No certificates ○ (E) ——
○ (B) Some certificates ○ (D) —— ○ (F) ——

**Solution:** Hashes don't provide integrity. An attacker can create a valid signature on their malicious public key just by hashing it.

Q3.6 (3 points) Instead of signing, use HMAC to create a certificate. For example, the root server signs the public key of `berkeley.edu` by applying `HMAC`($K$, `berkeley.edu`), where $K$ is the root's private signing key.

○ (G) All certificates ● (I) No certificates ○ (K) ——
○ (H) Some certificates ○ (J) —— ○ (L) ——

**Solution:** HMACs use symmetric keys, so there is no way for the signatures to be verified without knowing the server's secret key.

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

## Q4 (29 points)

Alice is using a DNS resolver to perform a DNS lookup for `www.google.com`. A single, valid nameserver is authoritative for each of the following zones:

| *Zone* | *Nameserver* |
|---|---|
| `.` | `a.root-servers.net` |
| `.com` | `a.gtld-servers.net` |
| `google.com` | `ns1.google.com` |

Assume no other legitimate clients will query the resolver (but the adversary can query it if they wish), the resolver's cache is initially empty, and the resolver uses iterative querying.

Assume that in DNSSEC, no one will accept a record unless it has a valid signature.

The attacker is on-path between the resolver and `ns1.google.com`, but off-path to the other name servers. The attacker also knows when Alice makes a request. **Assume DNS uses a static source port known to the attacker**.

For each part, select all of the records that the attacker can poison.

Q4.1 (4 points) Standard DNS is used.

- [x] (A) Alice's cached A record for `www.google.com`
- [x] (B) Resolver's cached NS record for `.com`
- [x] (C) Resolver's cached NS record for `google.com`
- [ ] (D) Resolver's cached NS record for `.`
- [ ] (E) ——
- [ ] (F) ——

**Solution:** The adversary can spoof the DNS response from the resolver to the client even though they are off-path since we are using vanilla DNS and the source port isn't randomized (poisoning Alice's A record). Furthermore, the adversary can mount a Kaminsky attack against the resolver by querying the resolver directly; since the attacker can predict the source port, this will be successful. So, they can poison the cache for all of the NS records except the root since this is hardcoded. The adversary can also use this method to poison the A record for www.google.com cached on the resolver, and when Alice queries it, the resolver will respond to the client with the poisoned cache entry.

Q4.2 (3 points) Standard DNS is used. Also, the resolver has a hardcoded NS record that maps the `google.com` zone to `ns1.google.com`, and a hardcoded A record with the IP address of `ns1.google.com`.

■ (G) Alice's cached A record for `www.google.com`

■ (H) Resolver's cached NS record for `.com`

☐ (I) Resolver's cached NS record for `google.com`

☐ (J) ——

☐ (K) ——

☐ (L) ——

**Solution:** Similar to above, the adversary can poison everything in the resolver's cache, except the hardcoded records.

Q4.3 (3 points) The resolver and nameservers use DNSSEC, and Alice uses standard DNS.

■ (A) Alice's cached A record for `www.google.com`

☐ (B) Resolver's cached NS record for `.com`

☐ (C) Resolver's cached NS record for `google.com`

☐ (D) ——

☐ (E) ——

☐ (F) ——

**Solution:** Same reasoning as above for the first option. The adversary can't poison the cache for any of the NS records since that would require forging a signature.

Q4.4 (3 points) The resolver and nameservers use DNSSEC, and Alice uses standard DNS. The adversary compromises `a.gtld-servers.net`.

■ (G) Alice's cached A record for `www.google.com`

■ (H) Resolver's cached NS record for `.com`

■ (I) Resolver's cached NS record for `google.com`

☐ (J) ——

☐ (K) ——

☐ (L) ——

**Solution:** Same reasoning as above for the first option. Controlling the `.com` domain allows the attacker to poison the resolver's cached NS record for `google.com` and `.com` since they have the private signing key and both of these domains are in-bailiwick. However, the fact that `.com` is in-bailiwick was considered outside the scope of the course and so this option wasn't graded.

Q4.5 (3 points) The resolver and nameservers use DNSSEC, and Alice uses standard DNS. The adversary compromises `ns1.google.com`.

■ (A) Alice's cached A record for `www.google.com`

☐ (B) Resolver's cached NS record for `.com`

■ (C) Resolver's cached NS record for `google.com`

☐ (D) ——

☐ (E) ——

☐ (F) ——

**Solution:** Controlling the `google.com` nameserver allows the attacker to poison the final result which they were already able to do. Same as the previous question, the attacker can also poison the cache for `google.com` but this option wasn't graded. Bailiwick rules stop them from poisoning the cache for higher zones.

Q4.6 (3 points) All parties use standard DNS, but the resolver and Alice encrypt their DNS messages with TLS.

■ (G) Alice's cached A record for `www.google.com`

■ (H) Resolver's cached NS record for `.com`

■ (I) Resolver's cached NS record for `google.com`

☐ (J) ——

☐ (K) ——

☐ (L) ——

**Solution:** The attacker can perform the Kaminsky attack to poison the `.com` + `google.com` NS records and on-path spoofing for the final DNS result.

Q4.7 (3 points) All parties use standard DNS, but Alice, the resolver, and `ns1.google.com` encrypt their DNS messages with TLS.

■ (A) Alice's cached A record for `www.google.com`

■ (B) Resolver's cached NS record for `.com`

■ (C) Resolver's cached NS record for `google.com`

☐ (D) ——

☐ (E) ——

☐ (F) ——

**Solution:** The attacker can use the Kaminsky attack to poison the NS records for the `.com` zone since the root nameserver and `.com` nameservers don't use TLS, and then can subsequently poison the `google.com` NS record and the final result.

Q4.8 (3 points) All parties use standard DNS, but everyone encrypts their DNS messages with TLS.

☐ (G) Alice's cached A record for `www.google.com`

☐ (H) Resolver's cached NS record for `.com`

☐ (I) Resolver's cached NS record for `google.com`

☐ (J) ——

☐ (K) ——

☐ (L) ——

**Solution:** The attacker hasn't compromised any of the nameservers so they can't do anything here. TLS prevents an on-path adversary from tampering with the messages between the parties.

Q4.9 (4 points) Alice and the resolver use standard DNS, but encrypt their DNS messages with TLS. The resolver and nameservers use DNSSEC.

☐ (A) Alice's cached A record for `www.google.com`

☐ (B) Alice's cached NS record for `google.com`

☐ (C) Resolver's cached NS record for `.com`

☐ (D) Resolver's cached NS record for `google.com`

☐ (E) ——

☐ (F) ——

**Solution:** The attacker can't poison any caches here due to DNSSEC, and can't compromise the client's connection with the resolver due to TLS.

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

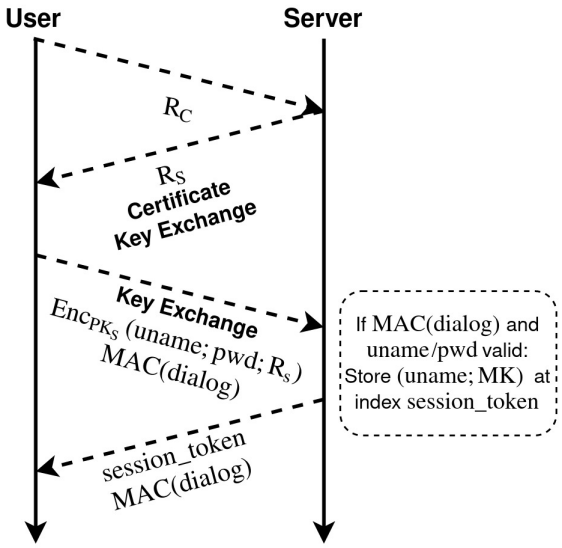
## Q5 (37 points)

*FastCash* is a fast banking service which requires users to log in before making a transfer, and uses TLS with ephemeral Diffie Hellman and RSA certificates to secure all their connections. They implemented a TLS extension called *0-Round Trip (0-RTT)* to speed up the connection process. 0-RTT changes the initial handshake as follows:

- Users authenticate themselves during the second round of the handshake
- If the user authenticates correctly, the server stores a session_token for that user

*(Recall that in TLS,* PS, $R_S$, *and* $R_C$ *generate a master key set* MK *which contains all the symmetric keys.* $Enc_{PK_S}$ *denotes RSA encryption using the server's public RSA key.)*

**A user only needs to perform the modified TLS handshake once**. To send an HTTP request after the initial connection ends, a user encrypts it using the keys derived in the initial handshake and attaches the session_token. The server verifies that the entry session_token : (uname, MK) exists and, if so, decrypts and executes the request as the user uname using the keys derived from MK.




Simplified diagram of modified initial TLS handshake

Assume that any on-path TCP injection attacks are impossible, and that once a user makes the initial modified TLS handshake, they will use the 0-RTT extension for future requests to the server.

Q5.1 (6 points) An on-path attacker observes an initial TLS handshake between a user and server, as well as a subsequent 0-RTT packet which contains an encrypted HTTP request. What can they do?

☐ (A) Read the user's future communications

■ (B) Break forward secrecy for that user's communications

☐ (C) Pretend to be the server to the user

☐ (D) Pretend to be the user to the server in a new handshake

■ (E) Replay the encrypted HTTP request to the server

☐ (F) Learn the master key set

**Solution:** The adversary can't learn any of the keys and so can't decrypt anything or fake being the server. While normally TLS doesn't authenticate the client, the 0-RTT extension involves authentication so without knowledge of the username/password the adversary can't pretend to be the user either. Including $R_S$ in the encryption stops the ciphertext from being replayed in a different session.
The adversary knows the session_token, so they can use the 0-RTT extension to replay an encrypted HTTP request they observed.
We gave credit for all answers for 'Break forward secrecy..' due to two separate interpretations of forward secrecy.

Q5.2 (6 points) Suppose we removed $R_S$ from the user's KeyExchange in the third step of the handshake. After observing an initial handshake between a user and the server, what can an on-path adversary do?

☐ (G) Read the user's future communications

■ (H) Break forward secrecy for that user's communications

☐ (I) Pretend to be the server to the user

■ (J) Pretend to be the user to the server in a new handshake

☐ (K) Learn the premaster secret

☐ (L) Learn the master key set

**Solution:** The adversary can't derive the premaster secret due to DH, and thus can't learn the master key set, violate forward secrecy, or learn future communications. Furthermore, they can't pretend to be the server to the user: the server's KeyExchange message, so the attacker can't modify or forge it, and consequently the adversary cannot predict the result of the Diffie-Hellman key exchange, the premaster secret, or the master key set.
However, the adversary can initiate a new handshake and replay the $\mathsf{Enc}_{\mathsf{PK}_S}(\mathsf{uname}; \mathsf{pwd})$ ciphertext observed in the first handshake to log in as the user.

Q5.3 (6 points) Due to a bug, an on-path adversary is able to choose the server's $R_S$. After observing an initial handshake between a user and the server, what can they do?

☐ (A) Read the user's future communications

■ (B) Break forward secrecy for that user's communications

☐ (C) Pretend to be the server to the user

■ (D) Pretend to be the user to the server in a new handshake

☐ (E) Learn the premaster secret

☐ (F) Learn the master key set

**Solution:** Same reasoning as above. The only thing that's different is the adversary has to force the server's $\mathsf{R_S}$ to be the same as used in the initial handshake to get the replay to work.

Q5.4 (6 points) An on-path adversary observes a user and the server communicating using 0-RTT for some time (without observing the initial handshake). At some point in the future, the adversary manages to learn all of the server's session_token : (uname, MK) entries. What can they do?

■ (G) Read the user's future communications

■ (H) Break forward secrecy for that user's communications

■ (I) Pretend to be the server to the user

☐ (J) Pretend to be the user to the server in a new handshake

☐ (K) Learn the premaster secret

■ (L) Learn the master key set

**Solution:** The adversary can learn the master key set (but not the premaster secret). This allows them to decrypt all future communications. Note that since we are essentially using a long-term private key (it is reused for all subsequent 0-RTT handshakes from the same user), we nullify the forward secrecy of using the Diffie-Hellman key exchange once the adversary has the master key set.

The adversary has no way to learn a valid ciphertext for the user's password so they can't pretend to be them. Since in future connections the user doesn't check the server's certificate, the adversary can pretend to be the server.

Q5.5 (10 points) Consider a MITM adversary during the initial handshake between a user and the server. Describe how this adversary can send a malicious HTTP request that appears to come from the legitimate user (Be specific with what is sent). Disregard any bugs from previous parts.

**Solution:** The adversary should do a DH MITM. When the server sends $\mathbf{R_S}$, the adversary should relay that same value on to the client. When the client sends the encrypted password, the adversary forces the client's connection to end by sending a RST packet. Next, the adversary replays this encrypted password to the server. This will be accepted by the server since $\mathsf{R_S}$ will be the same as the server was expecting, but replaces the client's $g^b$ value with the adversary's own $g^{b'}$. The adversary can then compute the result of the DH key exchange with the server, derive the premaster secret and master key set, and uses the derived MAC key to finish the handshake with the server. The adversary can now log in as the user using the session token returned by the server.
Normally TLS does not authenticate the client, so a MITM can always take over a connection initiated by the client. The key difference here is that the 0-RTT extension effectively authenticates the user. So the MITM can not only take over the connection, but also does so authenticated as the client; the server thinks messages are coming from the user, when actually they are coming from the adversary.

Q5.6 (3 points) Because of the vulnerability from the previous part, the company decides that it's too dangerous to allow all web pages to be accessible via 0-RTT. suppose they support the following three HTTP requests:

1. `GET` request for bank's homepage
2. `GET` request for bank's transfer page
3. `POST` request to execute a transfer

Below are different possible combinations of pages which will be made accessible via 0-RTT. Select the combination with no vulnerability or None if they are all vulnerable.

| | | |
|---|---|---|
| ◯ (G) 2, 3 | ◯ (I) 1, 3 | ◯ (K) —— |
| ● (H) 1,2 | ◯ (J) None | ◯ (L) —— |

**Solution:** We can never allow the POST request since this changes state. The question didn't specify how the webpages worked. If the webpages are static, and don't leak private information, then allowing the GET requests is fine. Otherwise, we can't allow those either. So 1,2 and None were accepted answers.

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

## Q6 (8 points)

Q6.1 (2 points) TRUE or FALSE: A NIDS always provides the most insight about ongoing network traffic.

◯ (A) True ● (B) False ◯ (C) —— ◯ (D) —— ◯ (E) —— ◯ (F) ——

**Solution:** False, a NIDS can't be used to monitor TLS traffic.

Q6.2 (3 points) An edgy hacker, xXOskiTheHackerXx, downloads a ransomware tool on GitHub and, without modifying it, tries to target the CDC. Which is the best detection strategy to detect this type of hacker?

● (G) Signature based
◯ (H) Behavior based
◯ (I) Anomaly based
◯ (J) Specification based
◯ (K) ——
◯ (L) ——

**Solution:** Signature based. The tools are public (on GitHub) and xXOskiTheHackerXx won't be able to modify the program to avoid signature detection.

Q6.3 (3 points) Andrew needs to decide between two burglar alarm systems - system A and system B. System A has a false positive rate of 0.05% and a false negative rate of 1%. System B has a false positive rate of 1% and a false negative rate of 0.05%.
The cost of a false positive is $100, because his parents fine him for causing a ruckus, and the cost of a false negative is $10000, because the burglar steals all his stuff. Which system should Andrew pick?

◯ (A) System A
◯ (B) System B
● (C) Not enough information
◯ (D) ——
◯ (E) ——
◯ (F) ——

**Solution:** Not enough information — we don't know how often attacks happen.

**This is the end of Q6. Proceed to Q7 on your Gradescope answer sheet.**

## Q7 (15 points)

Q7.1 (3 points) Alice clears all her network settings and broadcasts a DHCP discover message. What information should she expect to receive in the DHCP offer in response?

- ■ (A) DNS server
- ☐ (B) Source port
- ■ (C) Lease time
- ☐ (D) Premaster secret
- ■ (E) Gateway router
- ■ (F) IP address

**Solution:** The DHCP offer will include IP address, DNS server, gateway router, and how long the client can have these ("lease time"). The source port is determined by the user's machine. DHCP does not involve any premaster secret.

Q7.2 (6 points) After receiving the DHCP offer, Alice tries connecting to `www.cutecats.com`, but instead of pictures of cats, the site she gets is filled with dog photos.
How did the attacker compromise DHCP to accomplish this?

**Solution:** Since the DHCP discover message is broadcasted, any local attacker can hear the host's request. The attacker then spoofed the DHCP response by racing the actual server to send the DHCP offer to the client.

Which of the following could the attacker have replaced?

- ■ (G) DNS server
- ☐ (H) Source port
- ☐ (I) Lease time
- ☐ (J) Premaster secret
- ■ (K) Gateway router
- ☐ (L) IP address

**Solution:** Replacing the DNS server allows the attacker to redirect address lookups to a machine of the attacker's choosing. Replacing the gateway router allows the attacker to intercept all of the host's off-subnet traffic.

Q7.3 (3 points) Alice clears all her network settings and starts a new connection to `www.cutecats.com` with TCP. Now an off-path attacker wants to send a packet to the server to interfere with Alice's connection. What information do they need to know?

- ☐ (A) Server sequence number
- ■ (B) Source port
- ■ (C) Client sequence number
- ■ (D) Destination IP address
- ■ (E) Destination port
- ■ (F) Source IP address

**Solution:** An off-path attacker needs the IP addresses, ports, and sequence numbers to inject a packet. However, the server sequence number is not necessary because the server won't reject a packet with an incorrect ACK number.

Q7.4 (3 points) At some point, Alice's connection with `www.cutecats.com` is suddenly terminated. Assuming some information was leaked and the attacker correctly guessed the fields from the previous part, how was the attacker able to execute this attack?

◯ (G) —— ◯ (H) —— ◯ (I) —— ◯ (J) —— ◯ (K) —— ◯ (L) ——

**Solution:** The attacker injected a RST packet with the correctly guessed fields. This terminates the TCP connection.

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

## Q8 (18 points)

Q8.1 (4 points) Write a stateful firewall rule that would allow all TLS traffic from an external host `161.20.2.0` into your network `16.120.20.0/24`.

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

**Solution:** `allow tcp 161.20.2.0:* -> 16.120.20.0/24:*`
Common mistakes were not including the ports, including an incorrect port, forgetting to include the CIDR notation for `16.120.20.0/24`, specifying TLS as the protocol when a firewall would not have application layer context, etc.

Q8.2 (4 points) Recall that an attacker can spoof source IPs to hide themselves while executing a DoS attack. Assume the attacker securely randomly generates these IPv4 addresses. Describe a pattern in the packets that a network operator could observe to best discern whether or not their network is a victim of a DoS attack.

◯ (G) —— ◯ (H) —— ◯ (I) —— ◯ (J) —— ◯ (K) —— ◯ (L) ——

**Solution:** Look at the distribution of the source IP addresses of the incoming packets. If they are roughly uniformly distributed across the IP address space, this is likely to be the result of a DoS attack (see backscatter analysis).
Another viable option is to see that some source IP addresses are routed to private or non-routable IP addresses. Other accepted solutions mentioned the logic for maximum or minium sized packets.

Q8.3 (6 points) What intrusion detection method would be *best* fit to perform the previous analysis? Justify your answer.

◯ (A) HIDS ◯ (C) Logging ◯ (E) ——
● (B) NIDS ◯ (D) —— ◯ (F) ——

**Solution:** NIDS allows for real-time analysis, and by looking at the IP address source fields on the IP packets, there is no need for any visibility or context from the host. A NIDS is cheap to deploy.

Q8.4 (4 points) Describe a major drawback or exploit to the intrusion detection method you described above.

◯ (G) —— ◯ (H) —— ◯ (I) —— ◯ (J) —— ◯ (K) —— ◯ (L) ——

**Solution:** The NIDS could itself be overwhelmed by the volume of traffic. Also, if the bottleneck network link is upstream, the DoS attack might overwhelm that bottleneck link, causing many packets to be dropped before they reach the NIDS, making it harder for the NIDS to have full visibility of the attack.
Also accepted due to question ambiguity: a drawback of the intrusion detection method that is irrelevant in the context of DoS detection (e.g., traffic being encrypted).

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

## Q9 (12 points)

EvanBot has decided to switch career paths and pursue creating new cryptographic hash functions. EvanBot proposes two new hash functions, $E$ and $B$:

$$E(x) = H(x_1 x_2 \ldots x_{M-1})$$
$$B(x) = H(x_1 x_2 \ldots x_M || 0)$$

where $H$ is a preimage-resistant and collision-resistant hash function, $x = x_1 x_2 \ldots x_M$, $x_i \in \{0, 1\}$ and $||$ denotes concatenation.

In other words, $E(x)$ calls $H$ with the last bit of $x$ removed, and $B(x)$ calls $H$ with a 0 bit appended to $x$.

Q9.1 (3 points) Is $E(x)$ preimage-resistant? Provide a counter-example if it is not.

● (A) Yes ○ (C) — ○ (E) —

○ (B) No ○ (D) — ○ (F) —

Counterexample:

Q9.2 (3 points) Is $E(x)$ collision-resistant? Provide a counter-example if it is not.

○ (G) Yes ○ (I) — ○ (K) —

● (H) No ○ (J) — ○ (L) —

Counterexample:

**Solution:** $E(x)$ is preimage-resistant. Suppose not, i.e., given $E(x)$ we could find an $x'$ such that $E(x) = E(x')$. We will argue this means that $H$ is not preimage-resistant, either. Suppose we are given $H(y)$. Let $x = y0$, so that $E(x) = H(y)$. By assumption, we can find $x'$ such that $E(x) = E(x')$. Let $y' = x'_1 \cdots x'_{M-1}$. Then it follows that $H(y) = E(x) = E(x') = H(y')$, so given $H(y)$ we can find $y'$ such that $H(y) = H(y')$. This implies that $H$ is not preimage resistant. That is a contradiction, so our assumption that $E$ was not preimage-resistant must have been wrong.

$E(x)$ is not collision-resistant. Counter example: $E(1 \cdots 010) = E(1 \cdots 011)$,

Q9.3 (3 points) Is $B(x)$ preimage-resistant? Provide a counter-example if it is not.

● (A) Yes ○ (C) — ○ (E) —

○ (B) No ○ (D) — ○ (F) —

Counterexample:

Q9.4 (3 points) Is $B(x)$ collision-resistant? Provide a counter-example if it is not.

● (G) Yes
○ (H) No
○ (I) ——
○ (J) ——
○ (K) ——
○ (L) ——

Counterexample:

**Solution:**

$B(x)$ is preimage resistant, using the same reasoning as $E(x)$. (If there is an attack $B$'s preimage-resistance, then we can construct an attack against $H$'s preimage-resistance that succeeds half as often, which is often enough to show that $H$ is not preimage-resistant — but we were promised that $H$ is preimage-resistant, so it follows that $B$ must be preimage-resistant, too.)

$B(x)$ is collision-resistant. If $B(x)$ was not collision resistant, then we can find $x \neq y$ such that $B(x) = B(y)$. This can be rewritten as $H(x||0) = H(y||0)$. Letting $x' = x'||0$ and $y' = y'||0$, this means we found $x' \neq y'$ such that $H(x') = H(y')$, which proves that $H(\cdot)$ is not collision-resistant, which is a contradiction. Thus $B(x)$ must be collision-resistant.

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

## Q10 (12 points)

Alice comes up with a couple of schemes to securely send messages to Bob. Assume that Bob and Alice have known RSA public keys.

For this question, $Enc$ denotes AES-CBC encryption, $H$ denotes a collision-resistant hash function, $||$ denotes concatenation, and $\bigoplus$ denotes bitwise XOR.

Consider each scheme below independently and select whether each one guarantees confidentiality, integrity, and authenticity in the face of a MITM.

Q10.1 (3 points) Alice and Bob share two symmetric keys $k_1$ and $k_2$. Alice sends over the pair $[Enc(k_1, Enc(k_2, m)), Enc(k_2, m)]$.

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

☐ (B) Integrity ☐ (D) —— ☐ (F) ——

**Solution:** Note that $Enc$ denotes AES-CBC, not AES-EMAC, so we can only provide confidentiality. An attacker can forge a pair $[Enc(k1, c1), c1]$ given $[Enc(k1, c1||c2), c1||c2]$.

Q10.2 (3 points) Alice and Bob share a symmetric key $k$, have agreed on a PRNG, and implement a stream cipher as follows: they use the key $k$ to seed the PRNG and use the PRNG to generate message-length codes as a one-time pad every time they send/receive a message. Alice sends the pair $[m \bigoplus \text{code}, HMAC(k, m \bigoplus \text{code})]$.

■ (G) Confidentiality ■ (I) Authenticity ☐ (K) ——

■ (H) Integrity ☐ (J) —— ☐ (L) ——

**Solution:** This stream cipher scheme has confidentiality since the attacker has no way of coming up with the pseudorandomly generated one-time pads. $HMAC$ provides the integrity and authentication.

Q10.3 (3 points) Alice and Bob share a symmetric key $k$. Alice sends over the pair $[Enc(k, m), H(Enc(k, m))]$.

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

☐ (B) Integrity ☐ (D) —— ☐ (F) ——

**Solution:** Public hash functions alone do not provide integrity or authentication. Anyone can forge a pair $c, H(c)$, which will pass the integrity check and can be decrypted.

Q10.4 (3 points) Alice and Bob share a symmetric key $k$. Alice sends over the pair $[Enc(k, m), H(k||Enc(k, m))]$.

■ (G) Confidentiality ☐ (I) Authenticity ☐ (K) ——

☐ (H) Integrity ☐ (J) —— ☐ (L) ——

**Solution:** $H(k||Enc(k, m))$ is not a valid substitute for $HMAC$ because it is vulnerable to a length extension attack.

**This is the end of Q10. You have reached the end of the exam.**

# Fun Thing on Final Page

Here's a fish

><=>

Here's a phish

Congratulations, you are the 100,000th visitor to our website! Click here to claim your prize.

Here's a spearphish

<from no-reply@grapescope.com>

Hi Foo,

Your Midterm 2 for CS161, Spring 2020 has been graded! You can access your graded submission using the link below.

View your graded work

If you have never logged in to this site before, you can set your password. The login for your Grapescope account is `foo@bar.com`. (If you have any problems accessing the site, please contact help@grapescope.com.)

Statistics:
...