Weaver Fall 2017 — CS 161 Computer Security — Midterm 2

Midterm solutions updated May 2021 by CS161 SP21 course staff.

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

I am aware of the Berkeley Campus Code of Student Conduct and acknowledge that academic misconduct will be reported to the Center for Student Conduct.

SIGN your name: _____

PRINT your class account login: `cs161-`_____ and SID: _____

Name of the person sitting to your left: _____ Name of the person sitting to your right: _____

You may consult one sheet of paper of notes. You may not consult other notes, textbooks, etc. Calculators, computers, and other electronic devices are not permitted. We use Gradescope for grading so please write your answers in the space provided.

If you think a question is ambiguous, please come up to the front of the exam room to the staff. If we agree that the question is ambiguous we will add clarifying assumptions to the central document projected in the exam rooms.

You have 110 minutes. There are 5 questions, of varying credit (120 points total). The questions are of varying difficulty, so avoid spending too long on any one question. Use a #2/hb or softer pencil. For bubble questions, fill the bubble completely and clearly erase any mistakes.

Some of the test may include interesting technical asides as footnotes. You are not responsible for reading the footnotes.

Do not turn this page until your instructor tells you to do so.

Problem 1 ***True/False*** **(14 points)**

(a) (2 points) True/False: The origin policy for cookie access is different from the origin for JavaScript.

● True ○ False

Solution: True. Cookie policy and same-origin policy have different sets of rules.

(b) (2 points) True/False: An on-path attacker can disrupt any TCP connection the attacker can see.

● True ○ False

Solution: True. TCP messages are not secured by cryptography, so the attacker can read any messages sent over TCP. The on-path attacker can also see all the packet fields (including sequence numbers) and inject their own messages into the connection.

(c) (2 points) True/False: Without additional cryptographic authentication, conventional DNS is vulnerable to an on-path attacker.

● True ○ False

Solution: True. An on-path attacker can see the ID field (and source port, if source port randomization is in use) and spoof a malicious DNS response. If the malicious response reaches the victim before the legitimate response, the victim will accept the malicious response as valid.

(d) (2 points) True/False: Both ARP and DHCP can be spoofed by an attacker connected to the same WiFi network as the victim.

● True ○ False

Solution: True. ARP and DHCP both involve the victim broadcasting a message to everyone on the local network (e.g. the WiFi network) and accepting the first response. The attacker could spoof a malicious response to the victim, and if the malicious response reaches the victim before the legitimate response, the victim will accept the malicious response as valid.

(e) (2 points) True/False: Along with randomizing the source port and the identifier field, randomizing the destination port will further increase the entropy in preventing Kaminsky Attacks.

○ True ● False

Solution: False. The destination port in DNS is a fixed constant (port 53) and cannot be randomized. If the port was randomized, users would not know which port to send their DNS queries to.

(f) (2 points) True/False: Replacing a small set of input characters is generally sufficient to prevent CSRF attacks.

○ True ● False

Solution: False. CSRF attacks are mainly about the attacker tricking the victim into sending a request. CSRF does not involve malicious input being treated as code, so replacing input characters will not prevent CSRF attacks. (CSRF tokens are the main defense, and they don't involve replacing input characters.)

(g) (2 points) True/False: "SYN cookies" can work if the ACK is the first 4 bytes of $SHA256(SIP \| SPORT \| SEQ)$

○ True ● False

Solution: False.

Recall that when SYN cookies are in use, the server chooses its initial sequence number in the SYN-ACK packet such that the server sequence number encodes

any additional state needed to complete the handshake. This additional state usually is signed or HMAC'd to prevent tampering. The client must return this sequence number in the ACK before the server allocates some space to maintain the connection. This design lets the server avoid storing any state after receiving a SYN packet.

The ACK here doesn't work as a valid SYN cookie for several reasons. First, it only contains the first four bytes of a hash, which isn't enough to fully encode any information. Also, it's missing a random key that prevents an attacker from precomputing ACKs and sending them along with the initial SYN and circumventing the SYN cookie. You'd use HMAC instead because it's a keyed hash function that the attacker can't compute without the secret key.

Problem 2 ***Keep Your Answers Short and Tweet*** **(54 points)**

In all these questions please keep your answers short. If you can't fit it in roughly a tweet, you are probably writing too much.

(a) (4 points) Consider the following code snippet:

```
stmt = connection.prepareStatement("SELECT * FROM users
  WHERE USERNAME = ? AND ROOM = ?");
stmt.setString(1, username);
stmt.setInt(2, roomNumber);
stmt.executeQuery();
```

What type of attack does this type of coding defend against?

Solution: SQL Injection

Note the SQL query and the `prepareStatement` function. This code snippet is setting up a prepared statement, which is a defense against SQL injection.

(b) (4 points) A CA commonly validates certificates by checking whether the person requesting can add a piece of data onto the domain's web page. Does a CA's DNS server need to resist the Kaminsky attack?

Solution: Yes. Attacker could otherwise create a fake web page to host the check.

Specifically, an attacker can request a malicious certificate that states that the attacker's public key belongs to Google. The CA will ask the attacker to add some data on Google's homepage. The attacker uses the Kaminsky attack to fool the CA into thinking that the IP address of Google is an attacker-owned IP address. The CA sees the data on "Google's homepage" (actually the attacker's page) and issues the certificate.

(c) (4 points) In the name "`robert'; drop table students --`", what is the purpose of the `'`?

Solution: Terminates the string or closes the opening quote.

This SQL injection input is probably being placed in a SQL query that wraps the input around quotes. The single quote after `robert` acts as a closing quote to match the opening quote added by the query, which allows the rest of the input (the drop statement) to be treated as SQL code.

(d) (4 points) In the name "`robert'; drop table students --`", what is the purpose of the `--`?

Solution: Makes the rest of the statement a comment.

This SQL injection input is probably being placed in a SQL query that wraps the input around quotes. The two dashes at the end force the query to ignore the closing quote added by the query. (Otherwise, there would be an odd number of quotes: 2 added by the query, and one added by the input. The mismatched quotes would probably cause a syntax error.)

(e) (6 points) A page `foo.berkeley.edu` displays the value of the cookie "NAME" on the page `https://foo.berkeley.edu/xss` without any protection. You control the website `bar.berkeley.edu`. What is the domain, path, and flags you should set so *only* that page receives your value of name?

Solution: Domain: *.berkeley.edu (or berkeley.edu), path: /xss, Flags: Secure (can also add HTTPOnly but not essential)

(f) (4 points) `foo.berkeley.edu` wants to mitigate such cookie-based XSS attacks from other `berkeley.edu` sites. Why can't `foo`, without examining the content of the cookies themselves, distinguish between cookies set by `foo` and malicious cookies set by `bar`?

Solution: Because cookies are presented as name/value pairs. The domain and path are not presented (without examining the content of the cookie).

The browser allows multiple cookies to have the same name and value, as long as they have different domains and paths. `foo` and `bar` could each set a cookie with the same name and value, but different domains. (The browser usually stores the cookie under the unique identifier name/domain/path.)

(g) (4 points) `foo.berkeley.edu` wants to prevent clickjacking, but at the same time wants any other site to be able to embed `foo`. Why can't they prevent clickjacking?

Solution: Framebusting (preventing other websites from embedding `foo`) is a main defense against clickjacking. However, `foo` wants other sites to be able to embed it, so it can't use framebusting.

(h) (4 points) Why can't TLS protect against an on-path attacker who only wants to terminate connections?

Solution: Because TLS is built on TCP (i.e. the encrypted TLS packets are sent using TCP), and the on-path attacker can perform RST injection on the lower-level TCP connection to terminate the TLS connection.

(i) (4 points) Why can't TLS protect against a censor who wants to block specific websites?

Solution: TLS doesn't hide which website you're communicating with. TLS packets still need to contain information like the destination IP (in plaintext, with no encryption) so that lower-level protocols can send the encrypted message to the right destination.

Also note that in the TLS handshake, the server sends the certificate in plaintext, which would show the censor which website you're communicating with.

(j) (4 points) Why can't TLS protect against XSS attacks?

Solution: XSS vulnerabilities happen at at the application layer (HTTP/web), which is a higher layer than TLS.

In other words, TLS can secure communications between you and the server, but if the server itself is vulnerable, TLS does nothing to protect that.

(k) (4 points) What vulnerabilities can occur if a site renders part of the URL into the resulting web page?

Solution: Reflected XSS.

This is the definition of reflected XSS.

(l) (4 points) Why could a user site `user.github.com` steal a visitor's login cookies to `github.com`?

Solution: Because cookies can be read/set by subdomains. By the cookie policy, `user.github.com` can read and set cookies for `github.com`.

(m) (4 points) Why can't a user site `user.github.io` steal a visitor's login cookies to `github.com`?

Solution: By cookie policy, `user.github.io` cannot read or set cookies for `github.com` because the domains don't match.

Problem 3 ***The Internet of Shit*** **(12 points)**

A typical "Internet of Things" device has a webserver which people in the local network can use to manage it, reachable through `http://iosdevice.local/`. Of course this device, like most such devices, is horribly insecure, complete with a default username ("admin") and password ("secret") and has no other defenses against SQL injection, XSS attacks, CSRF attacks, etc. The URL encoding for ' is %27 : is %3A, < is %3C, > is %3F, space is %20 and / is %2F.

Lets consider some different ways of attacking it...

As an attacker, we can get a potential victim to visit our web page.

(a) (4 points) The login page is `http://iosdevice.local/login?user={USER}&password={PASSWORD}`. What "image" can we include on our page to ensure that a user who hasn't changed the password will be logged into the device?

Solution: A user who hasn't changed the password will have the default username (`admin`) and password (`secret`).

Including an image tag with a link causes the user's browser to automatically make a request to the link (trying to fetch an image). We want the user to be logged in, so we want to force their browser to make a request to the login URL with the default username and password:

```
<img src="http://iosdevice.local/login?user=admin&password=secret">
```

(b) (4 points) The following page `http://iosdevice.local/info?status={QUESTION}` includes the contents of `status` unescaped in the page. What iframe can we include on our page so that the script `http://evil.com/script.js` is run in the context of isodevice?

Solution:

```
<iframe src="http://iosdevice.local/info?
status=http%3A%2F%2Fevil.com%2Fscript.js">
```

This loads `http://evil.com/script.js` into the iframe.

Note that colons and slashes are URL-encoded because they're placed in a URL.

Note that script tags are not needed because this is not refected XSS–the question says that the contents of `status` are directly displayed in the page.

(c) (4 points) The following page `http://iosdevice.local/update?status={STRING}` contains an unprotected SQL request. If the attacker deletes the table `security` all security will be lost. What image can we include on our page to delete this table?

Solution: The classic SQL injection to delete a table is

```
' drop table security --
```

We URL encode the apostrophe and spaces, put the injection into the `status` argument, and place the link in an image tag:

```
<img src="http://iosdevice.local/update?
status=%27drop%20table%20security --">
```

Problem 4 ***TLS*** **(16 points)**

An attacker is trying to attack the company WoSlime and its users. Assume that users always visit WoSlime's website with an HTTPS connection, using RSA and AES encryption. (You may assume that WoSlime does not use certificate pinning) For each of the following attack scenarios, select all of the options that an attacker could achieve in that attack scenario.

(a) (4 points) If the attacker obtains a copy of WoSlime's private key, the attacker could:

- ● Impersonate the WoSlime web site to a user
- ● Measure the amount of traffic sent & received in a recorded connection between a user and WoSlime's website.
- ● Discover the plaintext of data sent during a recorded connection between a user and WoSlime's website.
- ● Inject content into a newly established connection between the user and Company's website that the attacker can observe as an on-path attacker.
- ○ Inject content into an established connection between the user and Company's website that the attacker can not observe as an on-path attacker.

Solution: Impersonate WoSlime: True. In the TLS handshake, WoSlime proves its identity by decrypting the premaster secret with its private key. If the attacker has WoSlime's private key, the user can't distinguish between WoSlime and the legitimate server.

Measure traffic: True. TLS does not hide the amount of traffic being sent. (The attacker doesn't need WoSlime's private key to do this. They just need to be able to observe the traffic.)

Discover plaintext: True. The attacker uses the private key to decrypt the premaster secret, derives the symmetric keys (using the premaster secret and the random nonces, which are sent in plaintext), and uses the symmetric keys to decrypt the recorded connection.

On-path injection: True. The attacker derives the symmetric keys (as described above) and then injects TCP packets containing content encrypted with the symmetric keys. (Recall that TLS is built on top of TCP, i.e. the encrypted messages are sent over TCP.)

Off-path injection: False. The attacker won't be able to see the TCP sequence numbers, so they can't inject content into the underlying TCP connection.

(b) (4 points) If the attacker obtains a copy of WoSlime's certificate, the attacker could:

- ○ Impersonate the WoSlime web site to a user
- ● Measure the amount of traffic sent & received in a recorded connection between a user and WoSlime's website.
- ○ Discover the plaintext of data sent during a recorded connection between a user and WoSlime's website.
- ○ Inject content into a newly established connection between the user and Company's website that the attacker can observe as an on-path attacker.
- ○ Inject content into an established connection between the user and Company's website that the attacker can not observe as an on-path attacker.

Solution: Recall that certificates are public, so obtaining a certificate doesn't give the attacker any additional abilities. Thus the attacker cannot impersonate the WoSlime website, decrypt data, or inject data into a TLS connection.

However, the attacker can still measure traffic, because TLS does not hide the amount of traffic being sent.

(c) (4 points) If the attacker obtains a copy of a trusted CA's private key, the attacker could:

- ● Impersonate the WoSlime web site to a user
- ● Measure the amount of traffic sent & received in a recorded connection between a user and WoSlime's website.
- ○ Discover the plaintext of data sent during a recorded connection between a user and WoSlime's website.
- ○ Inject content into a newly established connection between the user and Company's website that the attacker can observe as an on-path attacker.
- ○ Inject content into an established connection between the user and Company's website that the attacker can not observe as an on-path attacker.

Solution: The attacker can use the CA's private key to sign a malicious certificate claiming that the attacker is WoSlime (i.e. associating the attacker's private key with WoSlime). This will let the attacker impersonate WoSlime, because the attacker can present their malicious certificate to the user. The user will encrypt the premaster secret with the attacker's public key (as listed in the certificate), so the attacker can use their own private key to decrypt the premaster secret.

The CA's private key is not the same as the server's private key. Knowing the

CA's private key does not give the attacker any information about the actual server's private key, so the attacker cannot decrypt previously recorded data or inject data into a TLS connection.

The attacker can still measure traffic, because TLS does not hide the amount of traffic being sent. (The attacker doesn't need the CA's private key to do this. They just need to be able to observe the traffic.)

(d) (4 points) If the attacker obtains a copy of a trusted CA's certificate, the attacker could:

- ○ Impersonate the WoSlime web site to a user
- ● Measure the amount of traffic sent & received in a recorded connection between a user and WoSlime's website.
- ○ Discover the plaintext of data sent during a recorded connection between a user and WoSlime's website.
- ○ Inject content into a newly established connection between the user and Company's website that the attacker can observe as an on-path attacker.
- ○ Inject content into an established connection between the user and Company's website that the attacker can not observe as an on-path attacker.

Solution: The reasoning is similar to part (b). Recall that certificates are public, so obtaining a certificate doesn't give the attacker any additional abilities. Thus the attacker cannot impersonate the WoSlime website, decrypt data, or inject data into a TLS connection.

However, the attacker can still measure traffic, because TLS does not hide the amount of traffic being sent.

Problem 5 ***WPA3-PSK*** **(24 points)**

Outis made a horrible, horrible mistake[1]. In his general helpfulness, he volunteered to assist the IEEE in developing the WPA3-PSK standard. And now he has to evaluate alternative handshake schemes proposed to "securely" generate a key in the presence of rogue clients, rogue access points, and passive eavesdroppers.

As a reminder, a (slightly simplified) WPA2-PSK standard creates a PSK (Pre-Shared Key) as $PBKDF(pw, network-name)$. Then when handshaking the Access point selects a random value $ANonce$, broadcasting it to the client. The client then creates a random $SNonce$, calculates the keys as $H(ANonce||SNonce||PSK)$, and sends back $SNonce$ and $MIC(SNonce)$ (really a MAC but they name it differently). Since the only thing secret is the PSK, someone witnessing this handshake can attempt an off-line brute-force attack to find the password.

The first scheme Outis needs to evaluate, WPA3-DH, modifies this handshake using 3072-bit Diffie/Hellman. The protocol defines a P and g. The AP instead of $ANonce$ selects a random a and sends $g^a mod P$. The client selects a random b and calculates the keys as $H(g^{ab} mod P||PSK)$. The client returns $g^b mod P$ and $MIC(g^b mod P)$.

(a) (4 points) Does WPA3-DH prevent a passive eavesdropper from doing an offline brute force attack on the password? Why or why not? (A tweet-length answer please)

Solution: Yes: The attacker can't check if a guess is correct since they can't know the MIC key because its partially a function of the DHE.

Specifically, the attacker wants to guess a password, derive a PSK from their guess, calculate $H(g^{ab} mod P||PSK)$ in order to derive the symmetric keys, and check that the MIC matches. If the MIC matches, their guess is likely correct. However, the attacker cannot calculate $g^{ab} \bmod P$ from only $g^a \bmod p$ and $g^b \bmod p$ (because the discrete-log problem is hard), so they won't be able to perform this brute-force attack.

(b) (4 points) Does WPA3-DH prevent a passive eavesdropper who knows the password from decrypting connections? Why or why not? (A tweet-length answer please)

Solution: Yes: The DHE prevents this as well

Similar reasoning as above: even if the attacker knows the password, they can't calculate $H(g^{ab} mod P||PSK)$ and generate the symmetric keys because they can't calculate $g^{ab} \bmod P$.

[1] Having worked with standards committees himself, Nick could have warned Outis that this is thankless tasks that will make your eyes bleed and end in frustration as the "standard" becomes the worst combination of all proposals

(c) (4 points) Does WPA3-DH prevent a fake access point from gathering enough information to attempt an offline brute force attack? Why or why not? (A tweet-length answer please)

Solution: No: The attacker still gets enough info to try an offline brute-force attack.

The fake access point can perform a fake handshake with a victim as follows: Fake AP chooses an a and sends $g^a \bmod P$. Victim chooses a b and sends $g^b \bmod P$ and $MIC(g^b \bmod P)$, where the MIC uses a symmetric key derived from $H(g^{ab} mod P || PSK)$.

Now the attacker can brute-force the password as follows: guess a password and derive the PSK from their guess. Calculate $H(g^{ab} mod P || PSK)$ in order to derive the symmetric keys. Note that unlike the previous parts, the attacker can now calculate $g^{ab} \bmod P$ because the attacker selected a (and can thus calculate $(g^b)^a \bmod P$). Then check that the MIC sent by the victim matches the MIC calculated with the key derived from the password guess. If the MIC matches, their guess is likely correct.

The second scheme Outis needs to evaluate, WPA3-RSA, uses the PSK (a seemingly random value) to seed a pseudo random number generator to create a 3072b RSA private key K and a corresponding public key that is still kept secret. The AP sends $E_k(ANonce)$ (using RSA-OAEP) instead of ANonce, which the client can decrypt because it knows the PSK. The client sends back $SNonce$ and $MIC(Snonce)$ in the same way as the previous WPA2-PSK protocol.

(a) (4 points) Does WPA3-RSA prevent a passive eavesdropper from doing an offline brute force attack on the password? Why or why not? (A tweet-length answer please)

Solution: No: The attacker can use guesses of the PSK to find the RSA key

Specifically, the brute-force attack works as follows: guess a password and derive the PSK from their guess. Use the guessed PSK to derive the RSA private key. Use the guessed private key to decrypt ANonce. Derive the keys from the guessed ANonce, the guessed PSK, and the SNonce. See if the MIC calculated from the guessed derived keys matches the MIC observed in the handshake. If the MICs match, the guess is likely correct.

(b) (4 points) Does WPA3-RSA prevent a passive eavesdropper who knows the password from decrypting connections? Why or why not? (A tweet-length answer please)

Solution: No: The attacker knows the RSA key and can decrypt everything.

Use the password to derive the PSK. Then use the PSK to derive the RSA private key. Then use the RSA private key to decrypt ANonce. Use ANonce, SNonce, and the PSK to derive the symmetric keys. Use the symmetric keys to decrypt connections.

(c) (4 points) Does WPA3-RSA prevent a fake access point from gathering enough information to attempt an offline brute force attack? Why or why not? (A tweet-length answer please)

Solution: Yes: The attacker doesn't know the RSA key, and if you send "random" data, the OAEP will not be right on decryption (it pads things out with 0).

HOWEVER, because that wasn't clearly articulated that the client should actually check the OAEP padding bits as well, and it may not, you could just brute force assuming the client accepts things, so everyone gets full credit on this.