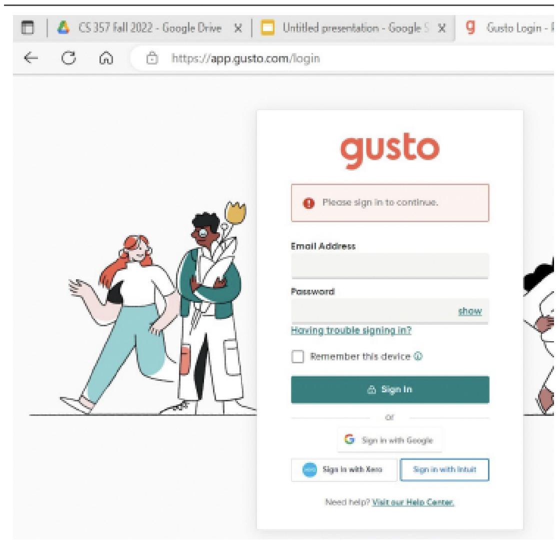
CAS CS 357

InClass Note 25

1. Logging into a website
    a. 
    CS 357 Fall 2022 - Google Drive
Untitled presentation - Google S
Gusto Login
https://app.gusto.com/login
gusto
Email Address
Password
show
Having trouble signing in?
Remember this device
Sign In
or
Sign in with Google
Sign in with Xero
Sign in with Intuit
Need help? Visit our Help Center.
    Please sign in to continue.
    b. First log in with username and password, then log in with our session cookie
    c. TLS doesn't provide all the security (security is combination of TLS and username and password and MFA and session cookie)
    d. TLS authenticates the server to the user
    e. This process is important because if we are not visiting the real website, we could be fished (might lead to stealth in login information)
2. SQL injection
    a. When typing username and password in the form → if it is not properly sanitized (put in little quote), error within server can occur
3. Dictionary attacks
    a. Strong password required → if we do not have a good password (password with enough randomness), because adversary can try a dictionary attack

   b. Dictionary attack → dictionary of passwords (try to enter the passwords one by one)
4. Credential stuffing
   a. Password reuse → if you have dark web and people sell passwords in the dark web → never use the favorite password in vulnerable websites (use secure and single use password) since adversaries can steal that password and try using on the bank account of that specific user
5. Password Hashing
   a. Password → we store passwords as hashes and we do not encrypt them
   b. Reason why we do not encrypt password → there needs to be the server storing the encrypted passwords needs to be able to decrypt those passwords in order to support logins → the risk is if adversary steals the passwords, they also steal the key → which would allow them to easily decrypt everything
   c. Passwords are stored as their hashes → when we log in with username and password → when we store password, we store the username, salt, and hash value (fast to look up which hash corresponds to which passwords → effectively reverse the password by first computing the dictionary of all hashes and then matching the hashes for the ones in the password file), salt makes the process harder because the salt would have to basically compute the dictionary for every single result → run through the entire computation with each salt (there is a lot of salt in the password file → one salt for one password → basically compute the entire table for every customer)
6. MFA

a. Adversaries are required to also steal the phone even if the password is leaked to the adversary (don't trust the password is good enough and therefore requires the physical holding of the device in MFA)

b. Phone based MFA?

   - Good protection but there's a risk that adversary can steal your phone number (not the physical phone) → SIM swapping(convince an employee to reassign to your physical phone) and SS7 attack (routing protocol to reaching a specific phone number)
   - Worry about phishing attack

c. TOTP based MFA?

   - Adversary needs physical access to the device assuming users do not fall for phishing attack
   - Fatigue attack → called over and over again to the extent where users get tired and press the on button (been phished from the adversary)
   - Phishing attack → adversary somehow tricking me to visit gust0.com instead of gusto.com and users type in username and password to the wrong site and adversary steals the information → scary thing is that adversaries could have a certificate for gust0.com (there may exist a TLS connection to confuse users)

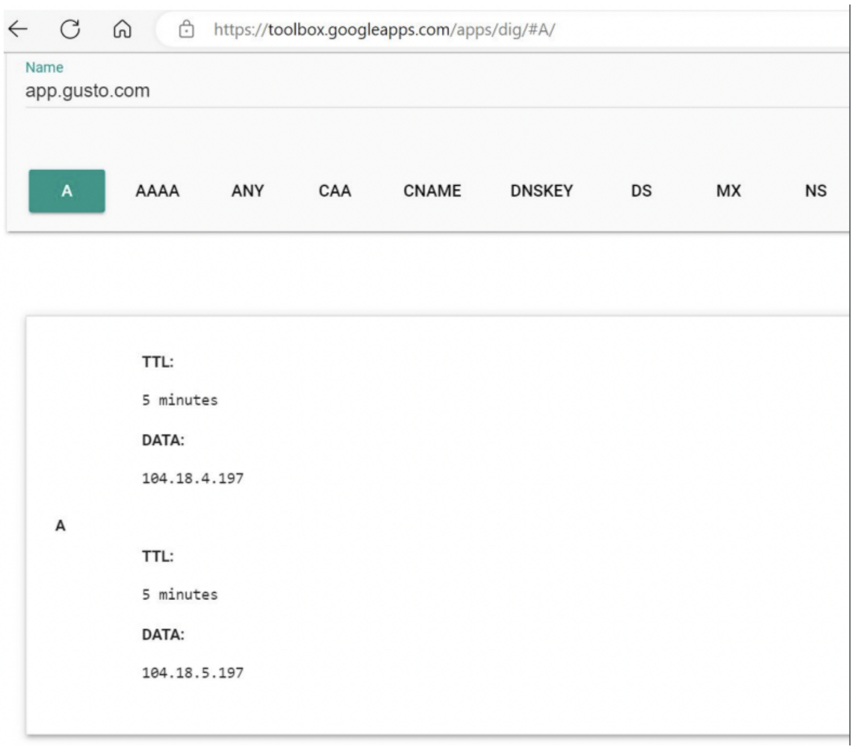
d. Webauthn based MFA?

   - Requires physical access to the device and it also prevents phishing attacks → still have fatigue attack problem if it uses push notifications (mostly being rolled out)

7. Once logged in, I stay logged in via a login cookie
    a. When logging in, type username and password → go to authentication server → if it looks okay → send back the login cookie → in the HTTP request coming back, there will be cookie = (for example L)
    b. Every time I access the website, I will be sending cookie L along with the HTTP header → keep record of the state that you logged in since it would be unnecessary to type the password every time you visit a different page within the same website (for example, email) → but these cookies will expire after some time → the process is repeated once logged out
    c. Can you send cookies in the clear over Http?
        - Cookies have a flag called secure flag (login cookie should have secure flag set) → if it does not, then it actually allows sending of the cookie over HTTP
    d. What happens if a man-in-the-middle adversary captures my login cookie? Is this possible if I am using TLS?
        - If we are using HTTPs and TLs, the adversary cannot see the cookie → gives us security since the cookies are encrypted
8. CSRF protection is needed while I'm logged in
    a. Example) Me and Alice → I visit 0.com → for some reason, a bad employee(insider attack) got me to click a link → pay the evil employee some money assuming that I am logged into gusto.com

b. Adversary creates a form that looks just like the form on gusto.com which includes javascript command of paying an evil person 100 dollars (make the form invisible so that users do not notice it)

9. When I visit a site, first I need to do a DNS lookup

a. 

https://toolbox.googleapps.com/apps/dig/#A/

Name
app.gusto.com

```
A
    TTL:
    5 minutes
    DATA:
    104.18.4.197

    TTL:
    5 minutes
    DATA:
    104.18.5.197
```

b. First step in gusto.com is DNS request

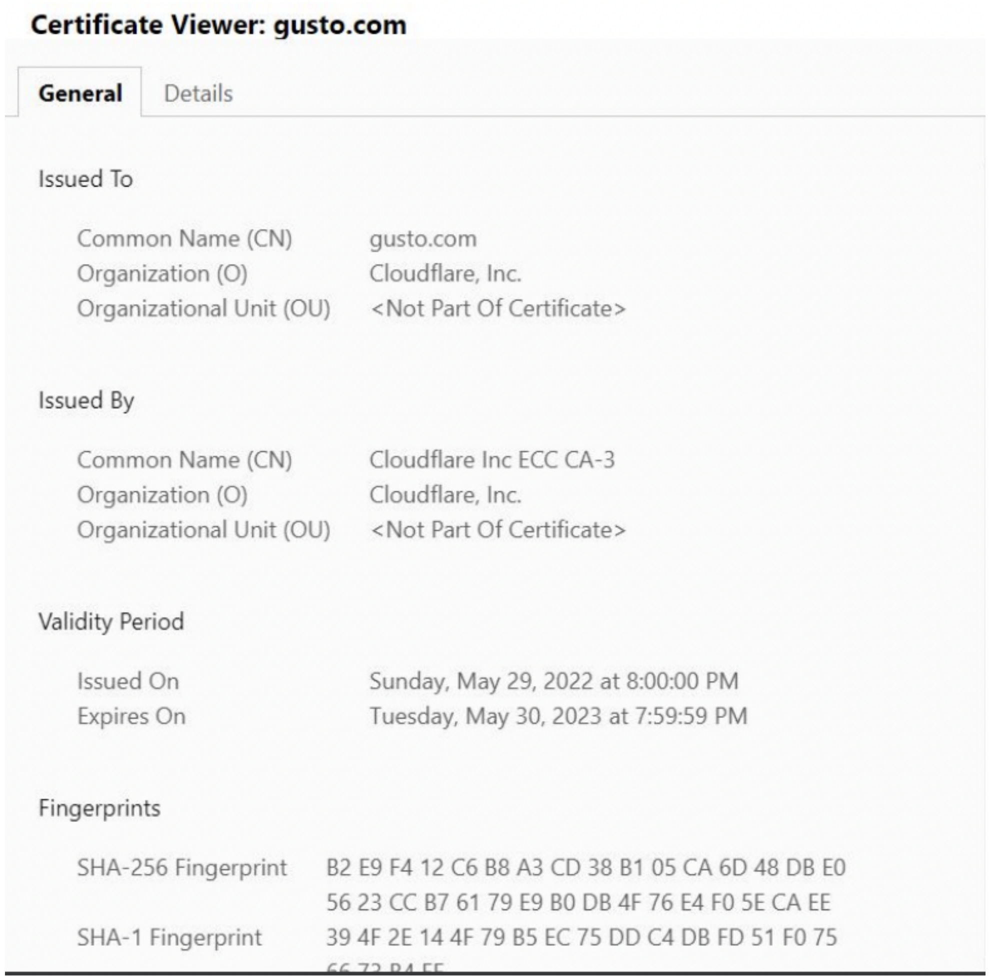
c. DNS tells the IP address of gusto.com so that we can visit the IP address of it to route to gusto.com

10. What happens if I get an invalid DNS response?

a. Ask for gusto.com and if I get back a malicious IP, 6.6.6.6 (malicious server gust0.com) from an adversary between the user and the recursive resolver → in this process, the user thinks that he/she is visiting gusto.com and the URL of the website actually states so (DNS request are typically not encrypted and authenticated, which allows such event to happen)

   b. Man-in-the-middle is created using DNS if we visit 6.6.6.6(thinking that it is real gusto.com) and the adversary, using the information we sent to the malicious server, sends it to the real gusto.com (user → adversary → gusto.com)

11. Assuming I get the correct IP address from DNS, I create a TCP connection to the gusto.com server
    a. If I do not get the correct IP address from DNS, as long as the certificate checking is done properly on the user side, this should not affect the connection
12. With TCP, it's hard for an off-path attacker to inject packets or disrupt communications. This is due to randomized initial sequence numbers
    a.
13. We use TLS to prevent on-pth or man-in-the-middle attackers
    a. When we open the website gusto.com → TLS connection is made
    b. When creating TLS connection, the server is authenticating to the client (TLS connection allows the users (us) to know that we are talking to the real gusto.com, not the adversary)
    c. The TLS (gusto) does not know who we are → to identify ourselves, we have to insert the correct username and password (but they have our IP address, which is meaningless)
    d. Therefore, when we use https and create a secure connection, all we are guaranteed to ourselves is that we know who the server is and we know that we are talking to the real server
    e. Once we log in, we use session cookies so that the server can identify the users
14. TLS certificate

a. 
Certificate Viewer: gusto.com

General | Details

Issued To

Common Name (CN)	gusto.com
Organization (O)	Cloudflare, Inc.
Organizational Unit (OU)	<Not Part Of Certificate>

Issued By

Common Name (CN)	Cloudflare Inc ECC CA-3
Organization (O)	Cloudflare, Inc.
Organizational Unit (OU)	<Not Part Of Certificate>

Validity Period

Issued On	Sunday, May 29, 2022 at 8:00:00 PM
Expires On	Tuesday, May 30, 2023 at 7:59:59 PM

Fingerprints

SHA-256 Fingerprint	B2 E9 F4 12 C6 B8 A3 CD 38 B1 05 CA 6D 48 DB E0 56 23 CC B7 61 79 E9 B0 DB 4F 76 E4 F0 5E CA EE
SHA-1 Fingerprint	39 4F 2E 14 4F 79 B5 EC 75 DD C4 DB FD 51 F0 75

b. When I visit gusto.com, I receive a certificate (actual certificate above)

c. There is the subject of the certificate (gusto.com in this case) and certificate authority is Cloudflare in this case and there is validity period

d. If we open the certificate up, we would also find the public key that belongs to gusto and also find a signature which is signed by secret key of Cloudflare

e. If secret key of Cloudflare is stolen → can have fraudulent certificate

f. Malicious certificate → subject is gusto.com, issuer is Cloudflare since the adversary stole the secret key → adversary choose a key sk_{evil} and pk_{evil} →

adversary sign the certificate using the secret key that he stole from Cloudflare → the user does not know that the secret key is stolen from the adversary

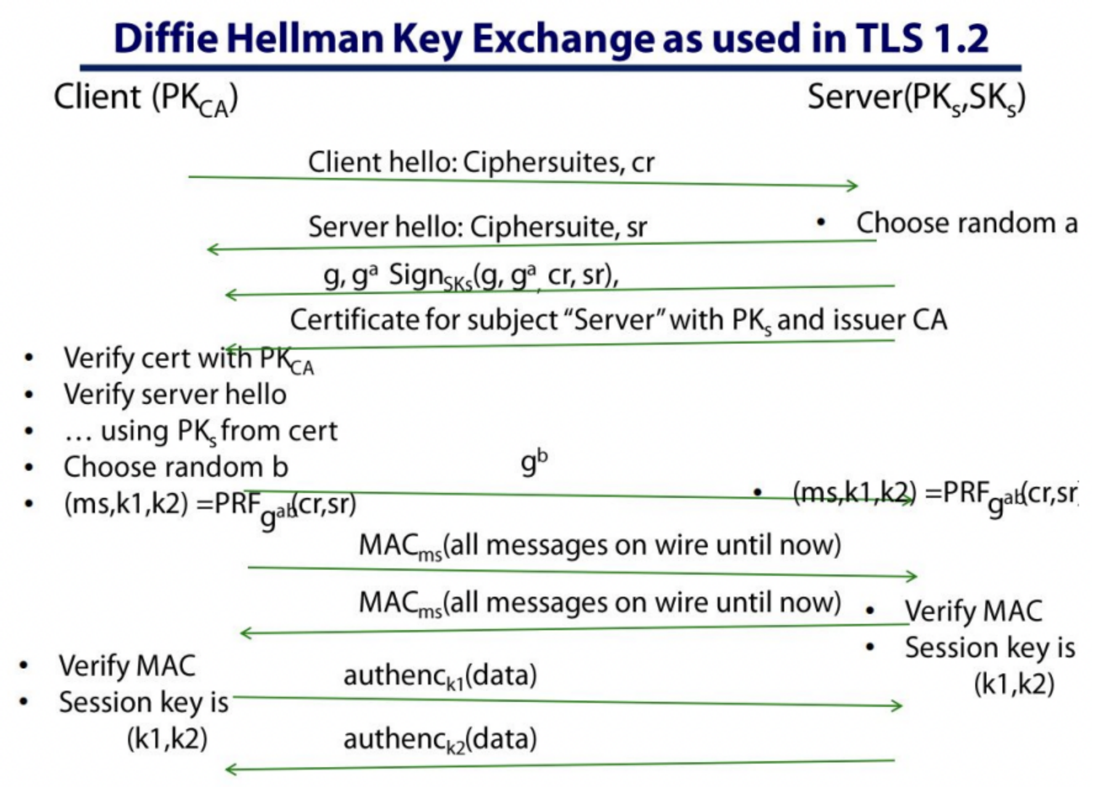
g. Danger occurs when two attacks combine → user visits the malicious server thinking that the user is visiting gusto.com → gets back fraudulent certificate on the path → adversary can create a TLS connection using the SK_{evil} as if the adversary is gusto.com → now the user is speaking what looks like secure TLS but the user is actually speaking to the adversary rather than the real gusto.com

h. TLS → server has its own keys (public and secret key that it uses to do the TLS handshake here(g^x and g^y and compute g^xy)

15. What happens if an adversary compromises the signing key of the certificate authority?

    a.

16. Now assume integrity of the Certificate Authority. What does the TLS handshake do?


Diffie Hellman Key Exchange as used in TLS 1.2
Client (PK_{CA})
Server(PK_s,SK_s)
Client hello: Ciphersuites, cr
Server hello: Ciphersuite, sr
• Choose random a
g, g^a $Sign_{SKs}$(g, g^a, cr, sr),
Certificate for subject "Server" with PK_s and issuer CA
• Verify cert with PK_{CA}
• Verify server hello
• … using PK_s from cert
• Choose random b
g^b
• (ms,k1,k2) =$PRF_{g^{ab}}$(cr,sr)
• (ms,k1,k2) =$PRF_{g^{ab}}$(cr,sr
MAC_{ms}(all messages on wire until now)
MAC_{ms}(all messages on wire until now)
• Verify MAC
• Session key is (k1,k2)
• Verify MAC
• Session key is (k1,k2)
$authenc_{k1}$(data)
$authenc_{k2}$(data)

    a. ____

b.

17. Recall: digital signatures

    a.

18. Recall: Message authentication codes

    a.

19. Recall: Symmetric Encryption

    a.