cookies and CSRF

intro to web

slides
bit.ly/cs161-disc

feedback
bit.ly/extended-feedback

hack of the day

- later in the slides!

general questions, concerns, etc.

cookies

cookies

- data used to maintain state across requests

cookies

- data used to maintain state across requests
  - HTTP is stateless
- created in one of three main ways
  - `Set-Cookie` header set by server's response
  - JavaScript in browser
  - manual creation by user (in browser)

cookies

- data used to maintain state across requests
  - HTTP is stateless
- created in one of three main ways
  - `Set-Cookie` header set by server's response
  - JavaScript in browser
  - manual creation by user (in browser)
- stored in the web **browser** (in a cookie jar)

cookies

cookies

- sending cookies:

cookies

- sending cookies:
  - browser automatically attaches relevant cookies with every request

cookies

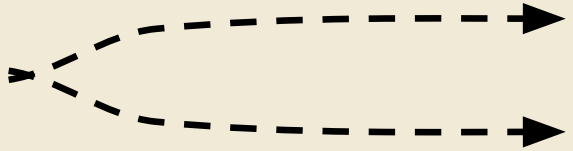
- sending cookies:
  - browser automatically attaches relevant cookies with every request
  - server uses received cookies to customize responses and connect related requests

parts of a cookie

Name	Theme
Value	Dark
Domain	toon.cs161.org
Path	/xorcist
Secure	True
HttpOnly	False
Expires	12 Aug 2021 20:00:00
(*other fields omitted*)	

parts of a cookie

data of the cookie

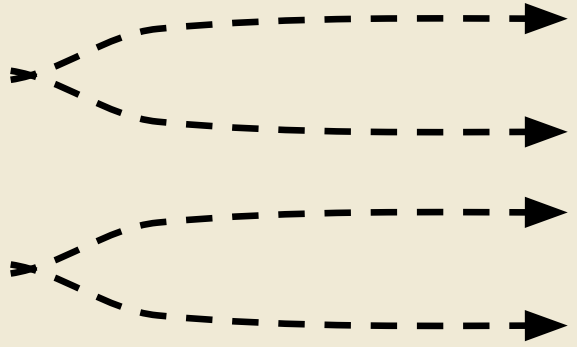


Name	Theme
Value	Dark
Domain	toon.cs161.org
Path	/xorcist
Secure	True
HttpOnly	False
Expires	12 Aug 2021 20:00:00
(*other fields omitted*)	

parts of a cookie

data of the cookie

what requests should this cookie be attached to?



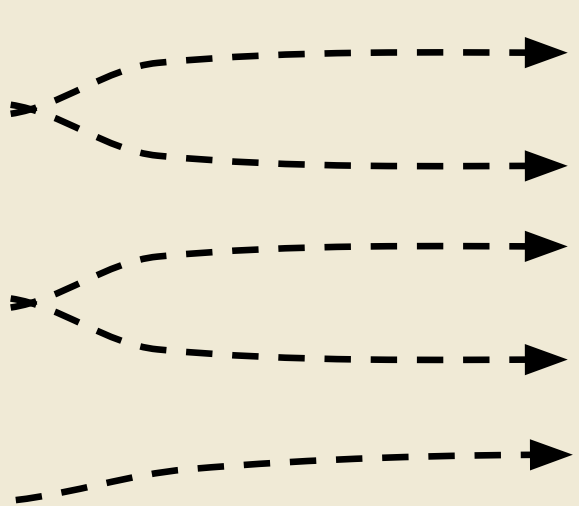
Name	Theme
Value	Dark
Domain	toon.cs161.org
Path	/xorcist
Secure	True
HttpOnly	False
Expires	12 Aug 2021 20:00:00
(*other fields omitted*)	

parts of a cookie

data of the cookie

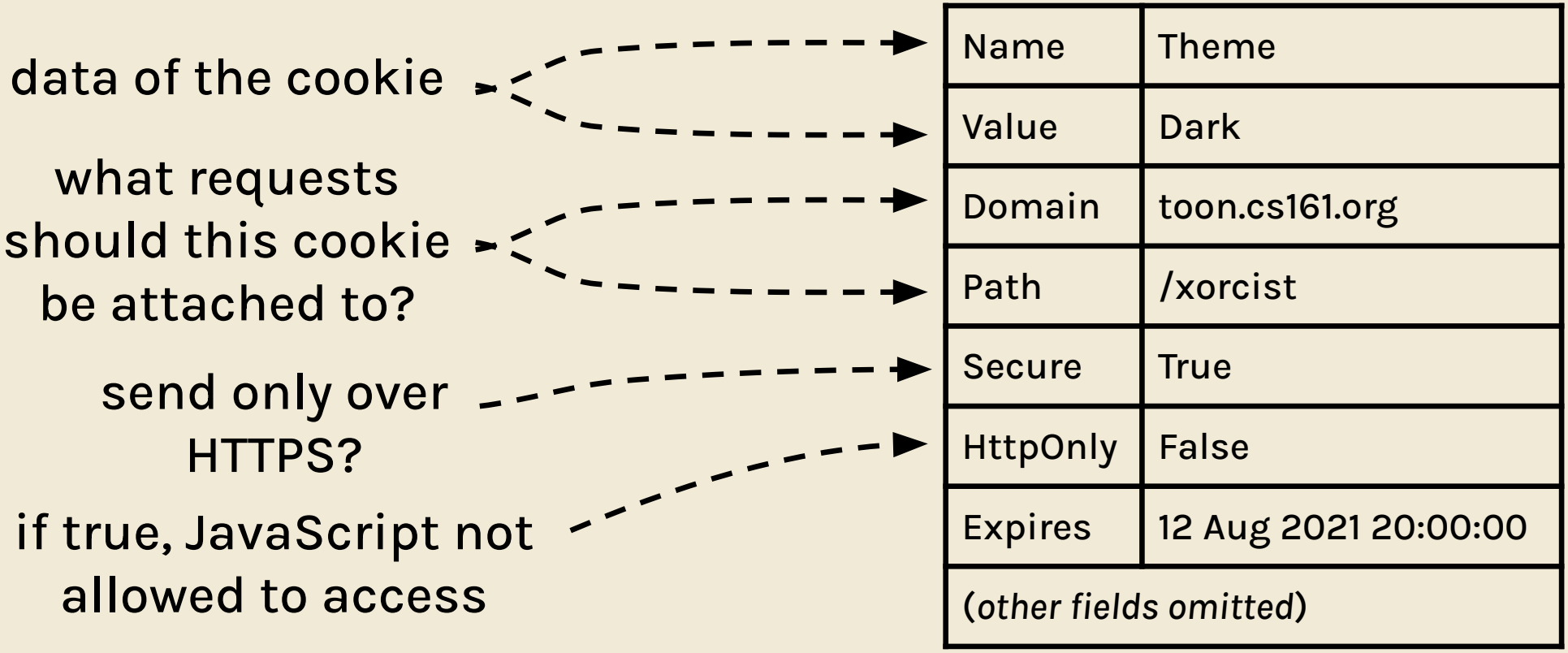
what requests should this cookie be attached to?

send only over HTTPS?

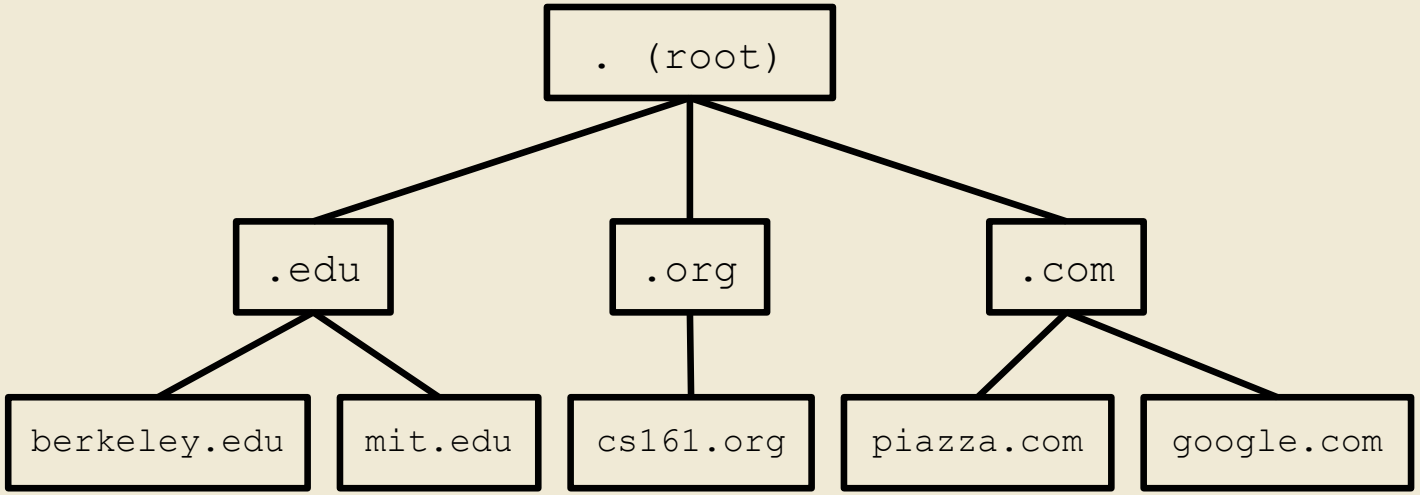


Name	Theme
Value	Dark
Domain	toon.cs161.org
Path	/xorcist
Secure	True
HttpOnly	False
Expires	12 Aug 2021 20:00:00
(*other fields omitted*)	

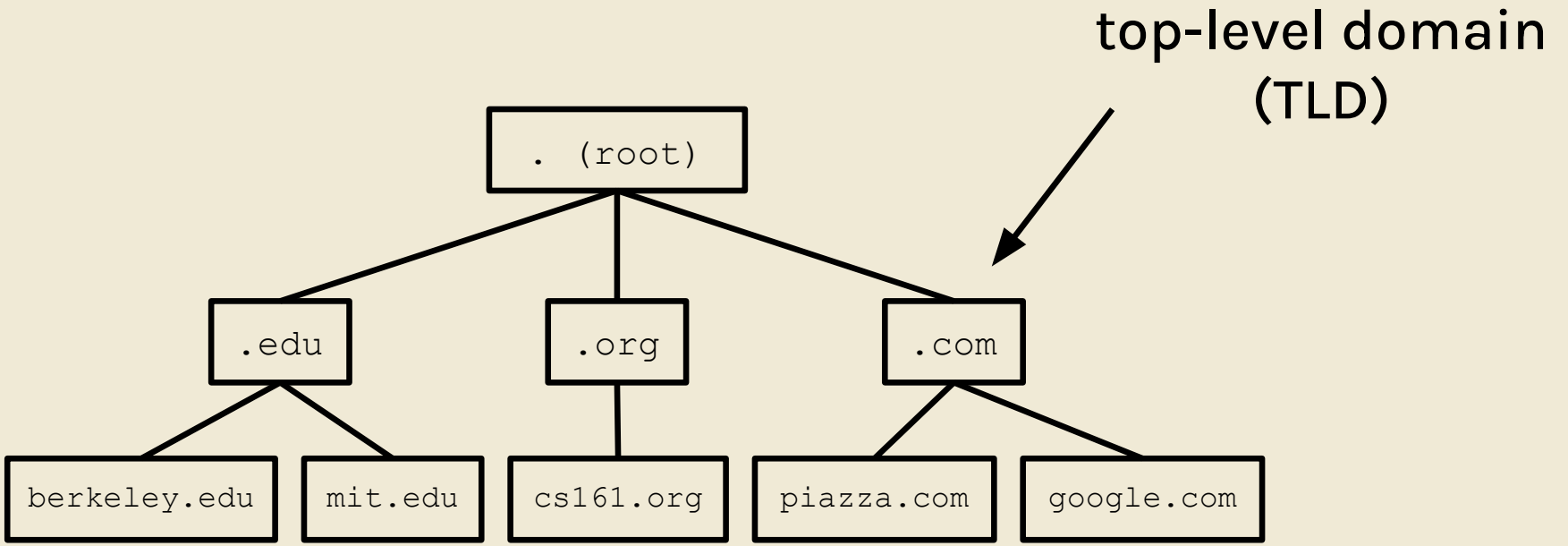
parts of a cookie



domain hierarchy



domain hierarchy



issues with cookies

issues with cookies

- servers should not be able to set cookies for unrelated websites
  - evil.com should not be able to set a cookie that gets sent to google.com

issues with cookies

- servers should not be able to set cookies for unrelated websites
  - evil.com should not be able to set a cookie that gets sent to google.com
- cookies shouldn’t be sent to the wrong websites
  - a cookie used for authenticating a user to Google should not be sent to evil.com

cookie policy

cookie policy

- aims to address issues

cookie policy

- aims to address issues
  - should a cookie be accepted when a server requests its creation?

cookie policy

- aims to address issues
  - should a cookie be accepted when a server requests its creation?
  - should a cookie be attached when the browser makes a request to a server?

cookie policy

- aims to address issues
  - should a cookie be accepted when a server requests its creation?
  - should a cookie be attached when the browser makes a request to a server?
- NOT same-origin policy

cookie policy

- aims to address issues
  - should a cookie be accepted when a server requests its creation?
  - should a cookie be attached when the browser makes a request to a server?
- NOT same-origin policy
  - dictates whether one page can access another (via JS or GET/POST requests)

cookie policy: setting cookies

cookie policy: setting cookies

- when server asks browser to create a cookie

cookie policy: setting cookies

- when server asks browser to create a cookie
- server with domain X can set a cookie with domain attribute Y if:

cookie policy: setting cookies

- when server asks browser to create a cookie
- server with domain X can set a cookie with domain attribute Y if:
  - X ends in Y

cookie policy: setting cookies

- when server asks browser to create a cookie
- server with domain X can set a cookie with domain attribute Y if:
  - X ends in Y
    - bcourses.berkeley.edu ends in berkeley.edu

cookie policy: setting cookies

- when server asks browser to create a cookie
- server with domain X can set a cookie with domain attribute Y if:
  - X ends in Y
    - bcourses.berkeley.edu ends in berkeley.edu
  - Y is not a top-level domain (TLD) (.com, .edu…)

cookie policy: setting cookies

- when server asks browser to create a cookie
- server with domain X can set a cookie with domain attribute Y if:
  - X ends in Y
    - bcourses.berkeley.edu ends in berkeley.edu
  - Y is not a top-level domain (TLD) (.com, .edu…)
- no restrictions on path

cookie policy: setting cookies

cookie policy: setting cookies

- can mail.google.com set cookies for Domain=google.com?

cookie policy: setting cookies

- can mail.google.com set cookies for Domain=google.com?
- can google.com set cookies for Domain=google.com?

cookie policy: setting cookies

- can mail.google.com set cookies for Domain=google.com?
- can google.com set cookies for Domain=google.com?
- can berkeley.edu set cookies for Domain=bcourses.berkeley.edu?

cookie policy: setting cookies

- can mail.google.com set cookies for Domain=google.com?
- can google.com set cookies for Domain=google.com?
- can berkeley.edu set cookies for Domain=bcourses.berkeley.edu?
- can google.com set cookies for Domain=com?

cookie policy: sending cookies

cookie policy: sending cookies

- when browser makes request to server

cookie policy: sending cookies

- when browser makes request to server
- browser sends cookie with domain attribute Y to server of domain X if:

cookie policy: sending cookies

- when browser makes request to server
- browser sends cookie with domain attribute Y to server of domain X if:
  - X ends in Y (Y is a suffix of X)

cookie policy: sending cookies

- when browser makes request to server
- browser sends cookie with domain attribute Y to server of domain X if:
  - X ends in Y (Y is a suffix of X)
  - path attribute of cookie is a prefix of the server's path

cookie policy: sending cookies

(server URL)

https://toon.cs161.org/cryptoverse/oneshots/subway.html

cs161.org/cryptoverse

(cookie domain) (cookie path)

- line up cookie domain and path with server

cookie policy: sending cookies

(server URL)

https://toon.cs161.org/cryptoverse/oneshots/subway.html

cs161.org/cryptoverse

(cookie domain) (cookie path)

- line up cookie domain and path with server
  - only send the cookie if they match!

cookie policy: sending cookies

(server URL)

https://toon.cs161.org/cryptoverse/oneshots/subway.html

cs161.org/cryptoverse

(cookie domain) (cookie path)

do I send the cookie?

cookie policy: sending cookies

(server URL)

https://toon.cs161.org/cryptoverse/oneshots/subway.html

cs161.org/cryptoverse

(cookie domain) (cookie path)

do I send the cookie?

yes!

cookie policy: sending cookies

(server URL)

https://toon.cs161.org/cryptoverse/oneshots/subway.html

cs161.org/exam

(cookie domain) (cookie path)

do I send the cookie?

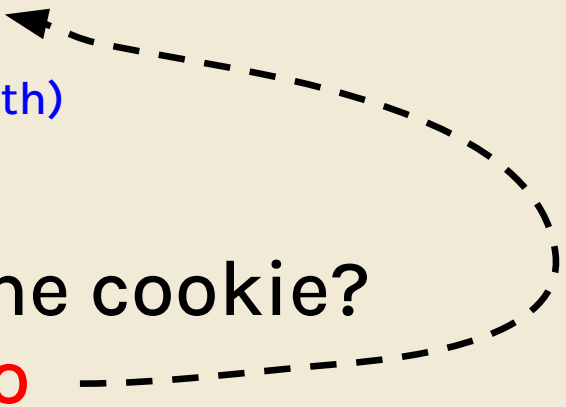
cookie policy: sending cookies

(server URL)

https://toon.cs161.org/cryptoverse/oneshots/subway.html

cs161.org/exam

(cookie domain) (cookie path)



cookie policy: sending cookies

(server URL)

https://bmail.berkeley.edu/mail

berkeley.edu/

(cookie domain) (cookie path)

do I send the cookie?

cookie policy: sending cookies

(server URL)

https://bmail.berkeley.edu/mail

berkeley.edu/

(cookie domain) (cookie path)

do I send the cookie?

yes!

cookie policy: sending cookies

(server URL)

https://boogle.com/bob

cool.boogle.com/bob

(cookie domain) (cookie path)

do I send the cookie?

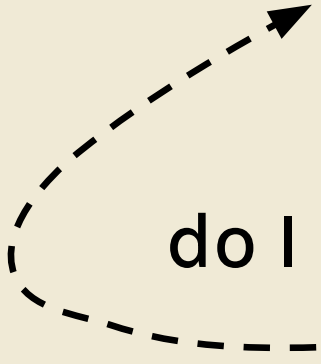
cookie policy: sending cookies

(server URL)

https://boogle.com/bob

cool.boogle.com/bob

(cookie domain) (cookie path)



do I send the cookie?

no

session tokens

session tokens

- cookies allow us to save webserver information

session tokens

- cookies allow us to save webserver information
- how can we save our login info?
  - prevents us from logging in each time

session tokens

- cookies allow us to save webserver information
- how can we save our login info?
  - prevents us from logging in each time
- session tokens!

session tokens: intuition

session tokens: intuition

- attending a concert:

session tokens: intuition

- attending a concert:
  - present ticket (username) and ID (password), receive a wristband (session token)

session tokens: intuition

- attending a concert:
  - present ticket (username) and ID (password), receive a wristband (session token)

- when you leave the venue, just show your wristband to get back in!

session tokens

session tokens

- when you log into a website, the server sets a cookie with your session token

session tokens

- when you log into a website, the server sets a cookie with your session token
- on future requests, browser automatically sends session token and server checks it

session tokens

- when you log into a website, the server sets a cookie with your session token
- on future requests, browser automatically sends session token and server checks it
- when you log out or the token expires, the browser and server delete the token

session tokens: security

session tokens: security

- what happens if an attacker steals the session token?

session tokens: security

- what happens if an attacker steals the session token?
  - attacker can make requests as someone else

session tokens: security

- what happens if an attacker steals the session token?
  - attacker can make requests as someone else
- browsers must enforce same origin policy to prevent stealing of session tokens

session tokens: security

- what happens if an attacker steals the session token?
  - attacker can make requests as someone else
- browsers must enforce same origin policy to prevent stealing of session tokens
- browsers should not send the session token to the wrong website

session tokens: security

- what happens if an attacker steals the session token?
  - attacker can make requests as someone else
- browsers must enforce same origin policy to prevent stealing of session tokens
- browsers should not send the session token to the wrong website
  - cookie policy

worksheet

(on 161 website)

CSRF

cross-site request forgery

cross-site request forgery

cross-site request forgery

- what if attacker tricks the victim into making an unintended request

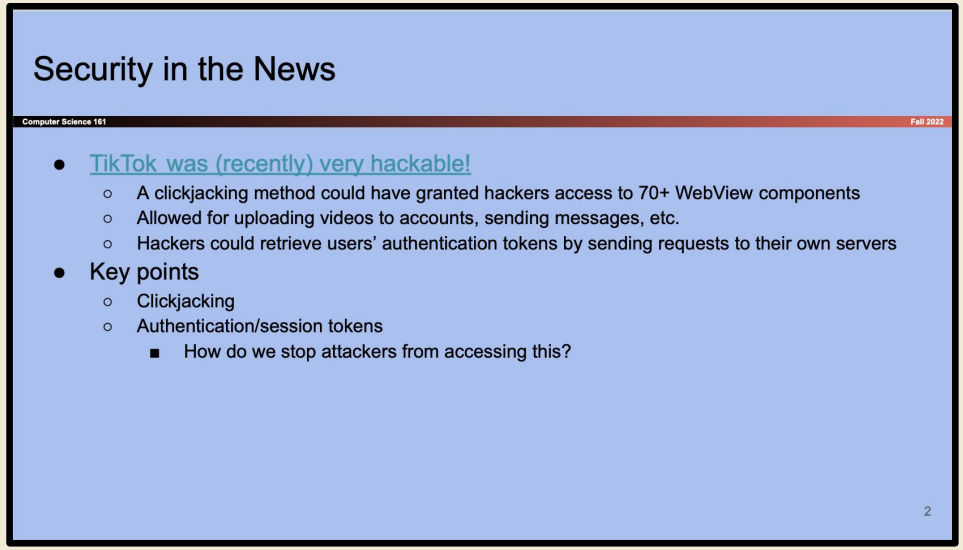
cross-site request forgery

- what if attacker tricks the victim into making an unintended request
  - browser attaches relevant cookies!

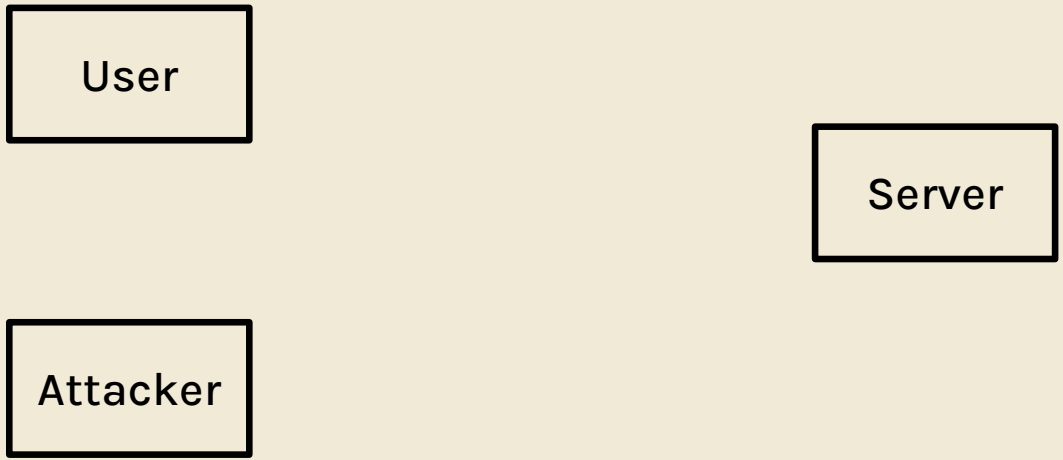
cross-site request forgery

- what if attacker tricks the victim into making an unintended request
  - browser attaches relevant cookies!

clickjacking used to make request as user

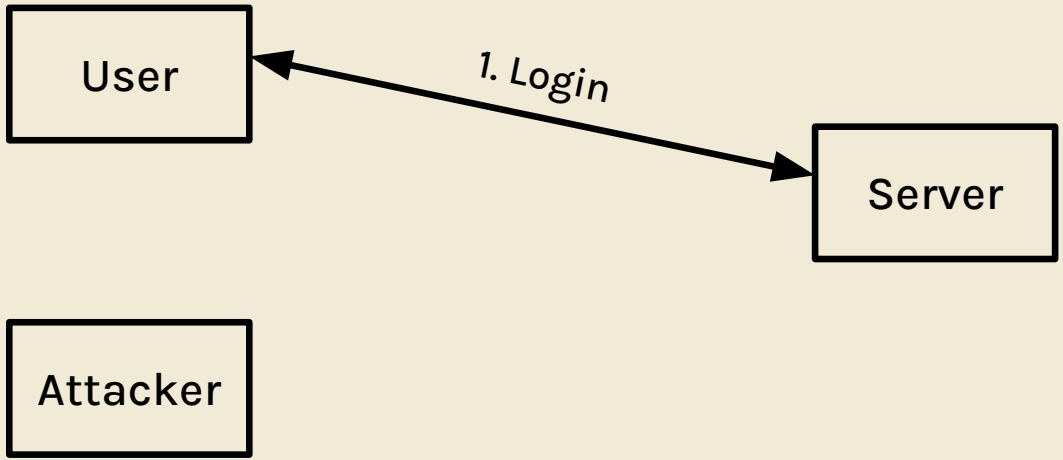


CSRF attack



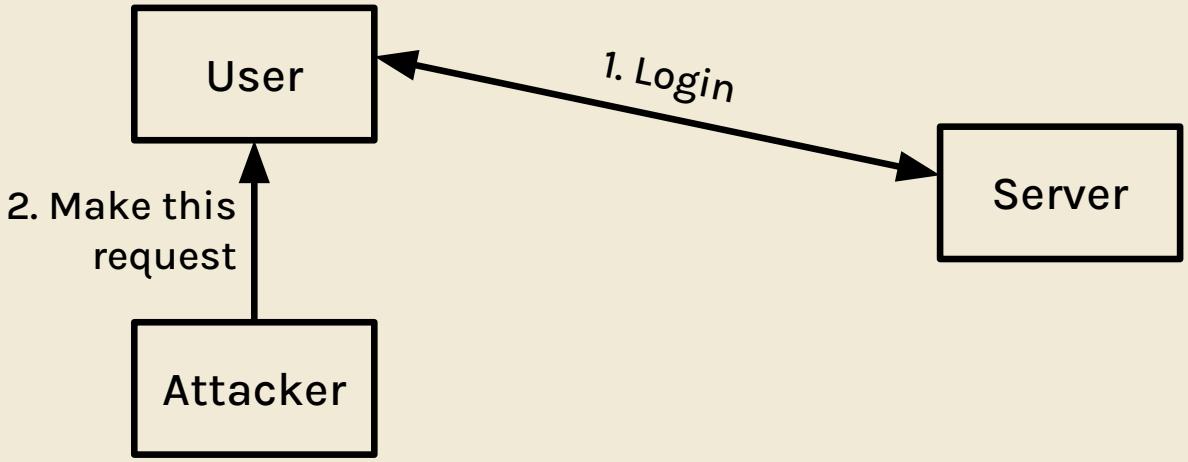
CSRF attack

1. user authenticates to the server
    - user receives cookie with valid session token



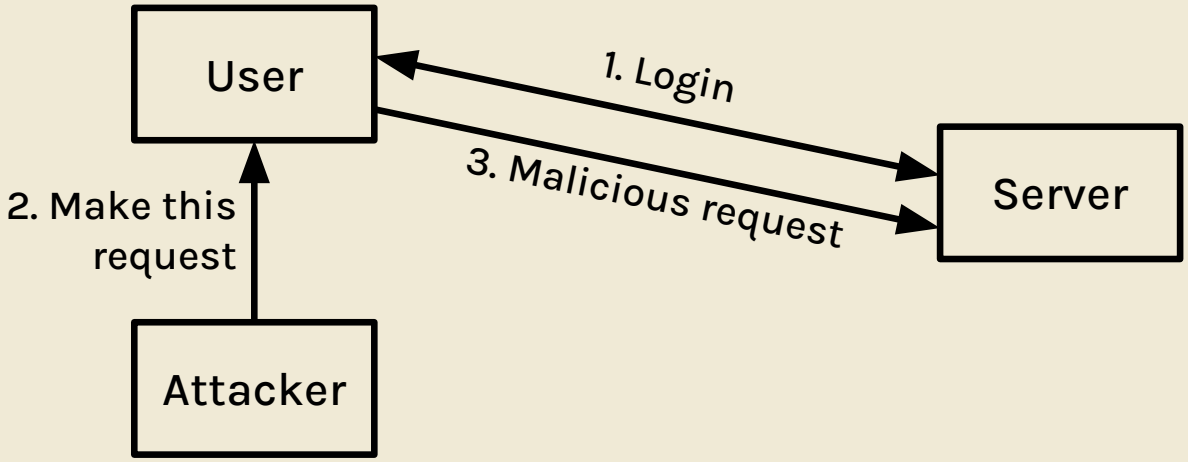
CSRF attack

1. user authenticates to the server
   - user receives cookie with valid session token
2. attacker tricks victim into making malicious request to server



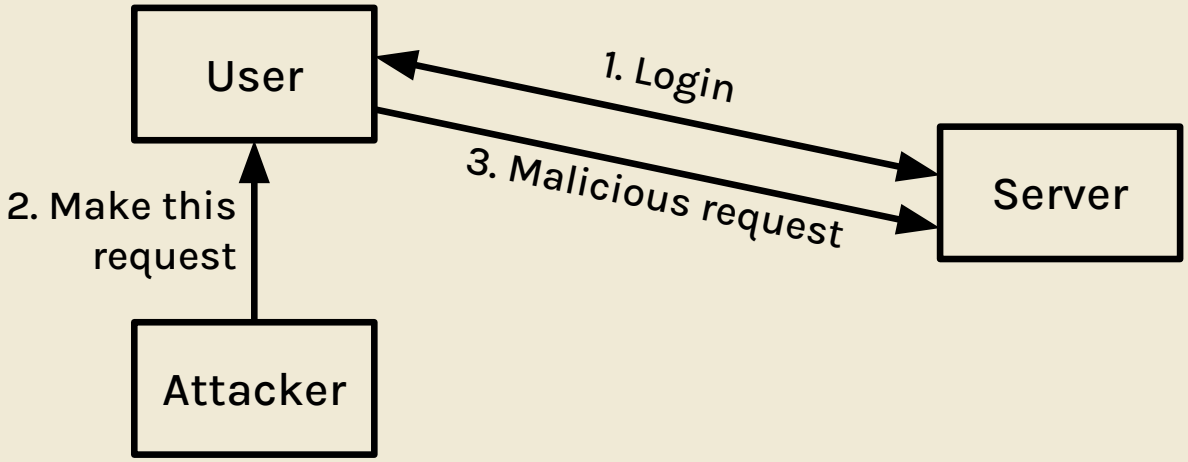
CSRF attack

1. user authenticates to the server
   - user receives cookie with valid session token
2. attacker tricks victim into making malicious request to server
3. server accepts malicious request from victim
   - recall: the cookie is automatically attached in the request



CSRF attack

1. user authenticates to the server
   - user receives cookie with valid session token
2. attacker tricks victim into making malicious request to server
3. server accepts malicious request from victim
   - recall: the cookie is automatically attached in the request



how?

strategies in executing CSRF: GET

strategies in executing CSRF: GET

1) trick the victim into clicking a link

strategies in executing CSRF: GET

1) trick the victim into clicking a link
   - direct GET request
     (https://www.bank.com/transfer?amount=100&to=Mallory)

strategies in executing CSRF: GET

1) trick the victim into clicking a link
   - direct GET request (https://www.bank.com/transfer?amount=100&to=Mallory)
   - link can open an attacker's website, which contains malicious JavaScript

strategies in executing CSRF: GET

1) trick the victim into clicking a link
2) put HTML on a website the victim will visit

strategies in executing CSRF: GET

1) trick the victim into clicking a link
2) put HTML on a website the victim will visit
    - put the HTML on a forum that accepts HTML

strategies in executing CSRF: GET

1) trick the victim into clicking a link
2) put HTML on a website the victim will visit
    - put the HTML on a forum that accepts HTML
    - <img src="https://www.bank.com/transfer?amount=100&to=Mallory">

strategies in executing CSRF: GET

1) trick the victim into clicking a link
2) put HTML on a website the victim will visit
    - put the HTML on a forum that accepts HTML
    - <img src="https://www.bank.com/transfer?amount=100&to=Mallory">
        - loading this “image” makes a GET request to the specified URL

strategies in executing CSRF: POST

strategies in executing CSRF: POST

1) trick the victim into clicking a link

strategies in executing CSRF: POST

1) trick the victim into clicking a link
   - makes a GET, so link should instead redirect to attacker's website with malicious JS

strategies in executing CSRF: POST

1) trick the victim into clicking a link
   - makes a GET, so link should instead redirect to attacker's website with malicious JS
2) put JavaScript on website the victim will visit

strategies in executing CSRF: POST

1) trick the victim into clicking a link
   - makes a GET, so link should instead redirect to attacker's website with malicious JS
2) put JavaScript on website the victim will visit
   - pay for a JS advertisement on a website

CSRF defenses

- CSRF tokens
- referer header
- SameSite cookie attribute

CSRF tokens

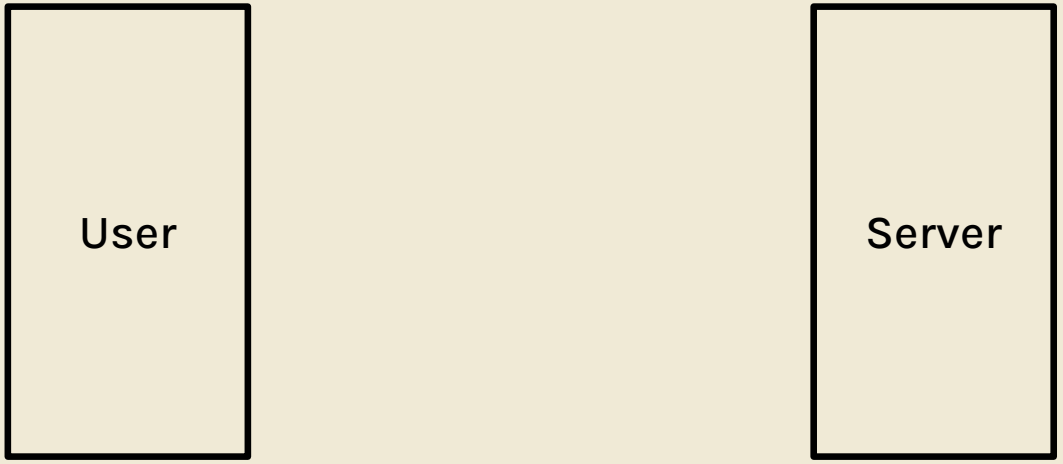
CSRF tokens

- generate a token unique to a user to be submitted along with a form/request

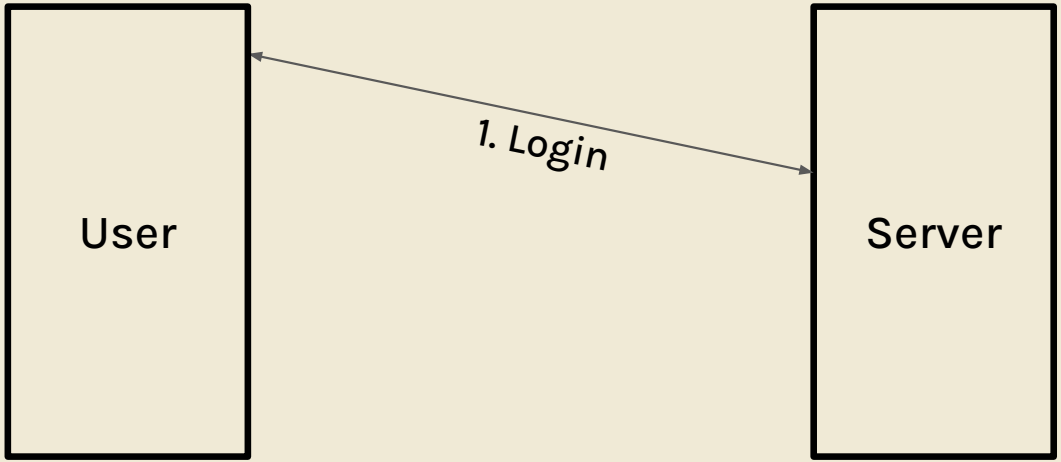
CSRF tokens

- generate a token unique to a user to be submitted along with a form/request
- stored in the HTML of the user's browser, cannot be seen by attacker

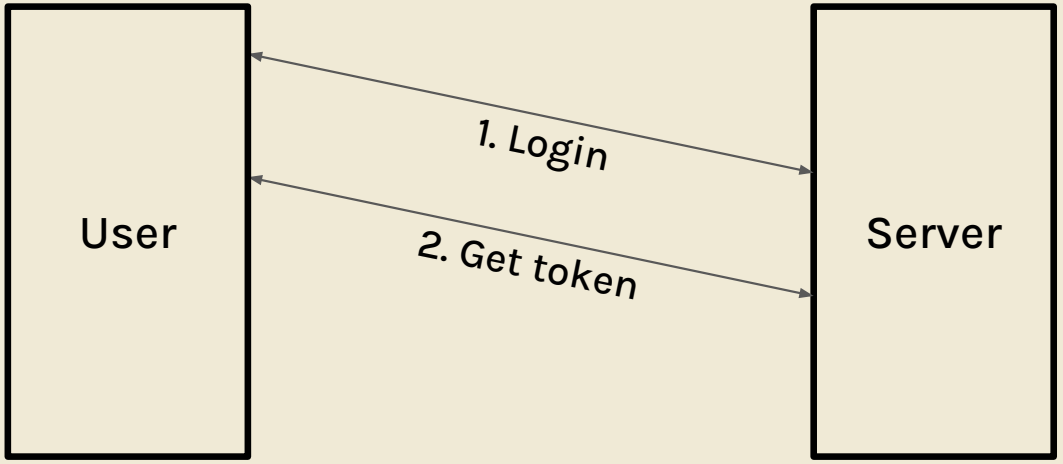
CSRF tokens



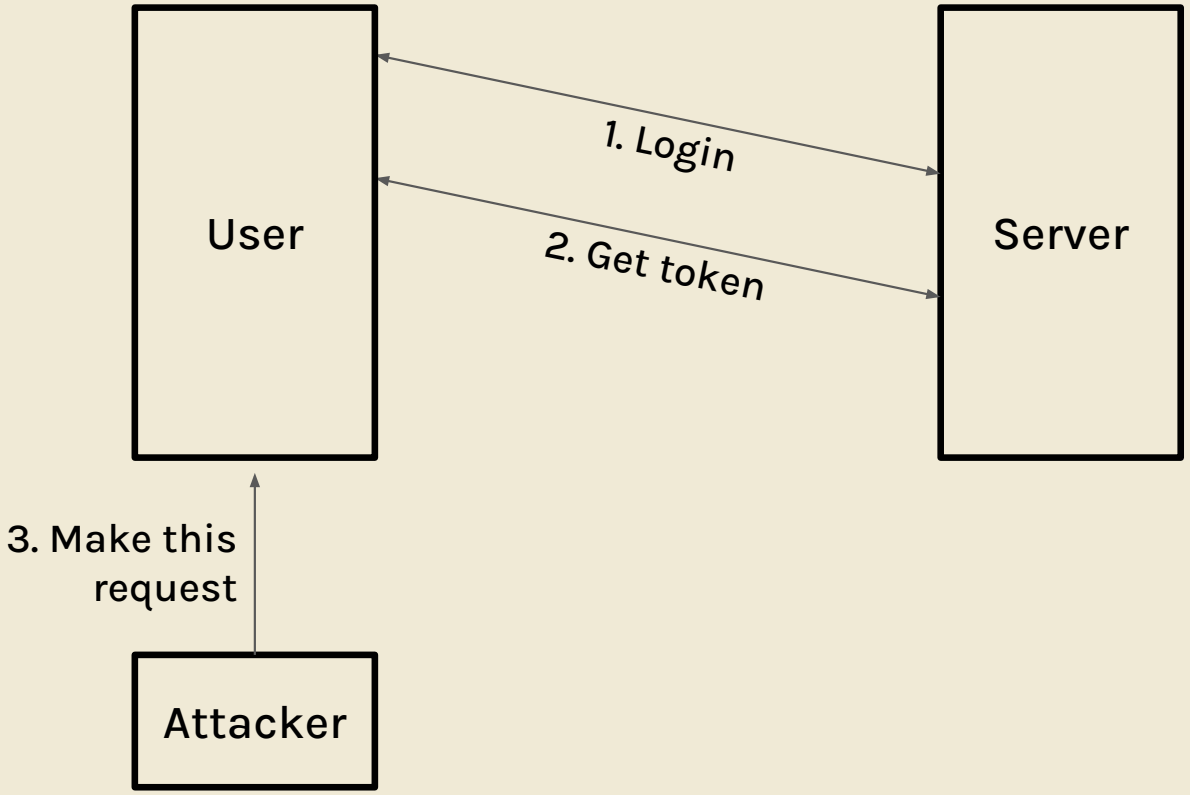
CSRF tokens



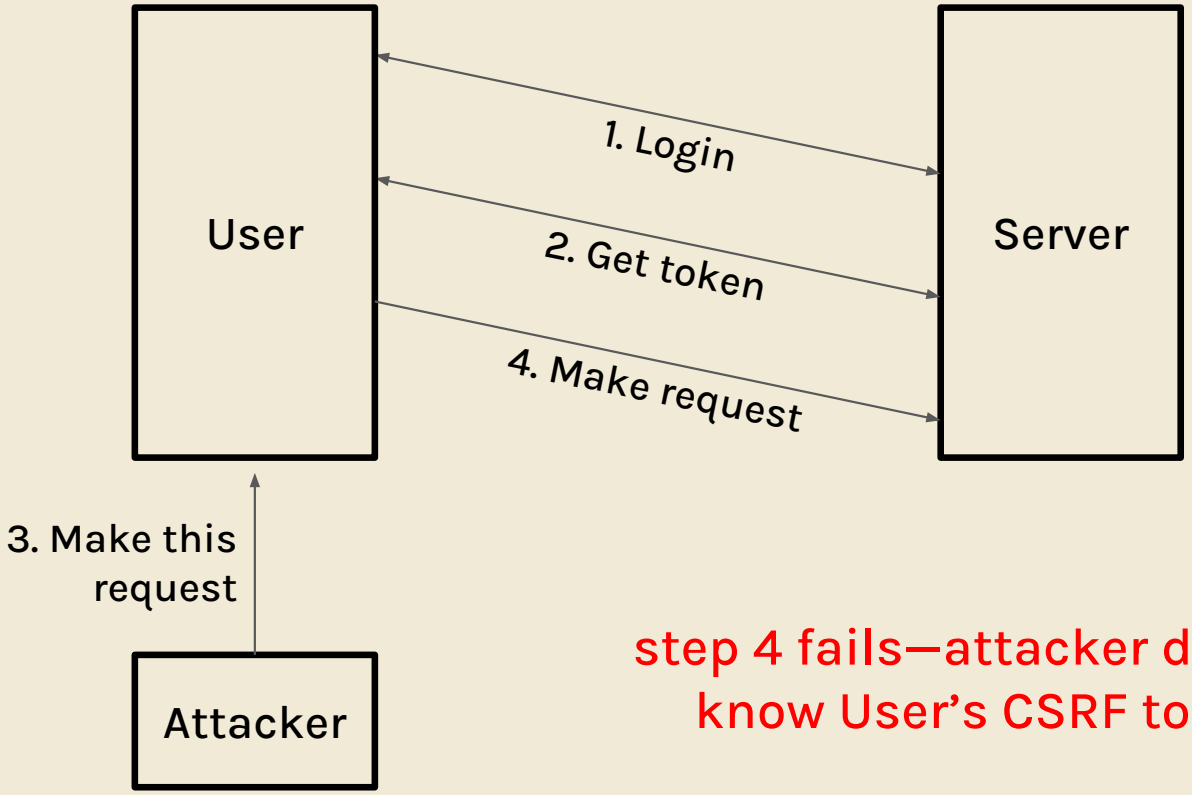
CSRF tokens



CSRF tokens



CSRF tokens



step 4 fails—attacker doesn't know User's CSRF token

referer headers

referer headers

- a header in an HTTP request indicating where the request came from

referer headers

- a header in an HTTP request indicating where the request came from
- CSRF attacker requests generally come from a different website

referer headers

- a header in an HTTP request indicating where the request came from
- CSRF attacker requests generally come from a different website
- idea: if request comes from a different website, don’t accept it

referer headers

- a header in an HTTP request indicating where the request came from
- CSRF attacker requests generally come from a different website
- idea: if request comes from a different website, don’t accept it
- issue: leaks where the request came from

referer headers

- a header in an HTTP request indicating where the request came from
- CSRF attacker requests generally come from a different website
- idea: if request comes from a different website, don’t accept it
- issue: leaks where the request came from
- issue: referer header is optional

referer headers

- a header in an HTTP request indicating where the request came from
- CSRF attacker requests generally come from a different website
- idea: if request comes from a different website, don’t accept it
- issue: leaks where the request came from
- issue: referer header is optional
  - what if blank or stripped by firewall?

SameSite cookie attribute

SameSite cookie attribute

- SameSite=strict attribute on cookie

SameSite cookie attribute

- SameSite=strict attribute on cookie
  - cookie only sent if cookie domain exactly matches origin domain

SameSite cookie attribute

- SameSite=strict attribute on cookie
  - cookie only sent if cookie domain exactly matches origin domain
  - if https://evil.com/ makes request to https://bank.com/transfer?to=mallory, cookies for bank.com not sent (origin domain is evil.com)

SameSite cookie attribute

- SameSite=strict attribute on cookie
  - cookie only sent if cookie domain exactly matches origin domain
  - if https://evil.com/ makes request to https://bank.com/transfer?to=mallory, cookies for bank.com not sent (origin domain is evil.com)
- issue: not implemented on all browsers

hack of the day

hack of the day

- Chromium bug bypasses SameSite cookie attribute

hack of the day

- Chromium bug bypasses SameSite cookie attr.
  - SameSite=Strict flag stops cookie from being sent by other websites

hack of the day

- Chromium bug bypasses SameSite cookie attr.
  - SameSite=Strict flag stops cookie from being sent by other websites
  - “Intent” URLs are when one app opens another—when Instagram opens a browser

hack of the day

- Chromium bug bypasses SameSite cookie attr.
  - SameSite=Strict flag stops cookie from being sent by other websites
  - “Intent” URLs are when one app opens another—when Instagram opens a browser
  - can use intent URLs to comply with SameSite=Strict

worksheet

(on 161 website)

feedback

bit.ly/extended-feedback

slides: **bit.ly/cs161-disc**