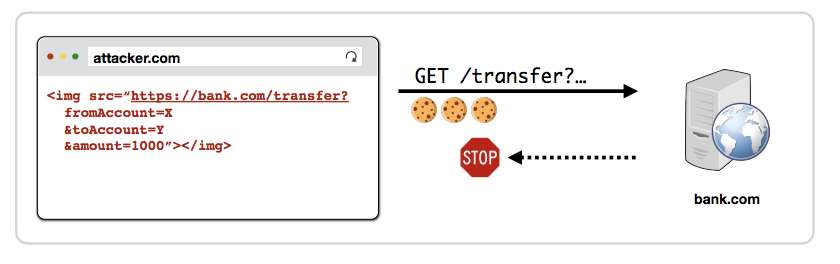
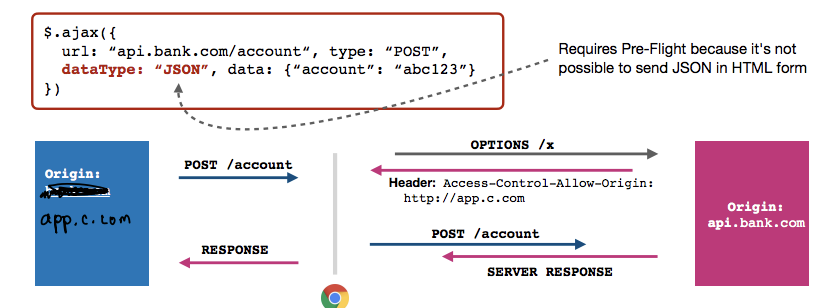
CAS CS 357

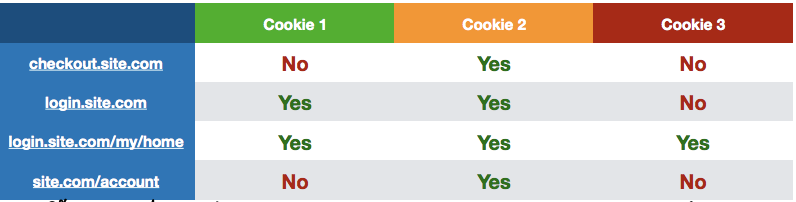
In-Class Note 6

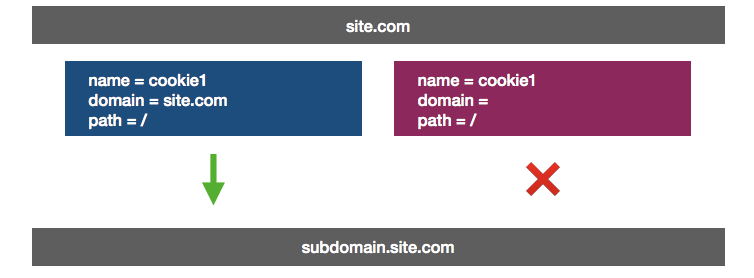
1. DOM Same Origin Policy
2. Websites can include resources from any web origin but the requesting website cannot inspect content from other origins
3. 
4. Requesting website cannot see what came back from bank.com since they have different origin
5. An origin is defined as a (http vs https scheme, domain, port)
6. Ex) http, standford.edu, 80
7. DOM SOP vulnerabilities (such as CSRF)
8. This can pose security risk because attackers might not need to view response to request to pull off their attack
9. 
10. Nothing stops one website from getting information from another website
11. Same origin policy does not stop making requests between webpages
12. JavaScript Requests
13. JavaScript can make new requests for additional data and resources
14. 
15. Cross-Origin Resource Sharing (CORS)
16. By default, JavaScript cannot read data sent back by a different origin
17. 
18. Servers can add Access-Control-Allow-Origin (ACAO) header that tells browser to allow access to content to be read by another origin
19. ACAO is like allow list where it enforces same origin policy except the allow list that the website allows and if it matches, bank.com can manipulate information
20. However, it is important to not allow every websites and allow only trustable websites since the defense of same origin policy disappears here
21. Simple vs Pre-Flight Requests
22. When request would have been impossible without JavaScript, CORS performs a Pre-Flight Check to determine whether the server is willing to receive request from the origin
23. 
24. If request does not go through HTML, you cannot pass request because there is no header
25. Pre-Flight Check creates an artificial header by requesting ahead of time to api such as api.bank.com and by checking whether the header is included in ACAO
26. HTTP Cookies
27. “In scope”, cookies are sent based on origin regardless of requester
28. Cookie Same Origin Policy
29. Cookies use a different definition of origin than the DOM: (domain, path): (cs155.standard.edu, /foo/bar)
30. A page can set cookie for its domain or any parent domain
31. Can set cookie for its path or any parent path
32. Browser sends cookies that are in URL’s scope
33. Cookies that belong to domain or parent domain AND are located at the same path or parent path
34. Setting Cookie Scope
35. Websites can set scope to be any parent of domain and URL path
36. Cs155.standford.edu can set cookie for cs155.stanford.edu (O)
37. Cs155.stanford.edu can set cookie for stanford.edu (O)
38. Stanford.edu cannot set cookie for cs155.stanford.edu (X)
39. Website.com/ can set cookie for website.com/ (O)
40. Website.com/login can set cookie for website.com/ (O)
41. Website.com cannot set cookie for website.com/login (X)
42. Scoping Example
43. name = cookie1 name = cookie2 name = cookie3

value = a value = b value = c

domain = login.site.com domain = site.com domain = site.com

path = / path = / path =/my/home

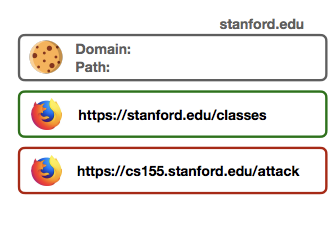
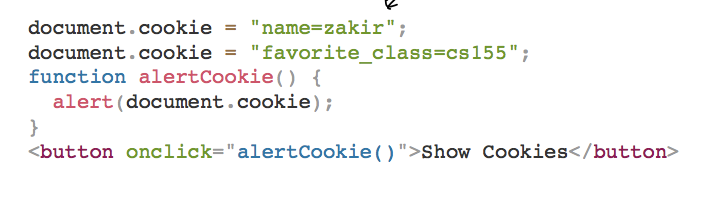
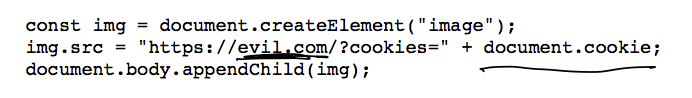
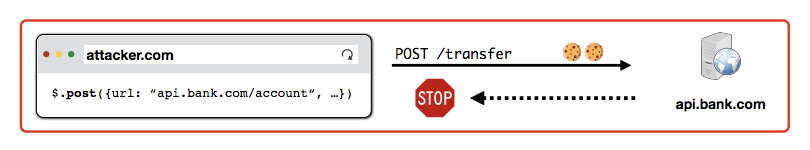


1. No Domain Cookie
2. Most websites do not set Domain. In this situation, cookie is scoped to the exact hostname the cookie was received over and is not sent to subdomains
3. 
4. Cookie Scoping
5. Example Cookie:

Set-cookie: id=a3fWa; Domain=Stanford.edu

1. If domain is set in cookie, then cookie will be sent to subdomain matches

For example, cs155.stanford.edu

1. If no Domain is set in cookie, the cookie will be sent to only exact domain matches (no subdomains)
2. If Path is not set in a cookie, then it defaults to the current document path
3. All subdirectories in path are sent the cookie
4. If you want all pages on site to receive a cookie set at /login, then you need to set Path=/
5. 
6. JavaScript Cookie Access
7. Developers can additionally manipulate in-scope cookies through JavaScript by modifying
8. 
9. Cross-site scripting attack comes in here
10. Xxx.site.com can have HTML that includes JavaScript which can lead to further attacks using the JavaScript code since it always runs with permissions of window
11. In other words, JavaScript can send cookie to attacker as part of the code and send it to other places to attack the user
12. Third Party Access
13. If your bank includes Google Analytics JavaScript, can it access your Bank’s authentication cookie?
14. Yes. JavaScript always runs with permissions of window
15. 
16. JavaScript has access to cookies
17. Images, in the other hand, does not run with permissions of the page (same origin policy)
18. HttpOnly Cookies
19. You can set setting to prevent cookies from being accessed by document.cookie API
20. It prevents Google Analytics from stealing your cookie –
21. Never sent by browser to Google because (google.com, /) does not match (bank.com, /)
22. Cannot be extracted by Google JavaScript that runs on bank.com
23. Set-Cookie = id=a3fWa; Expires = Thu, 21 Oct 2021; HttpOnly
24. Secure Cookies
25. Set-Cookie = id=a3fWa; Expires = Thu, 21 Oct 2021; Secure
26. Secure cookie is only sent to server with encrypted request over HTTPS protocol
27. Cross-Site Request Forgery (CSRF)
28. Cross-site request forgery (CSRF) attacks are a type of web exploit where website transmits unauthorized commands as user that web app trusts
29. 
30. In CSRF attack, user is tricked into submitting an unintended web request to a website
31. Preventing CSRF Attacks
32. Cookies do not indicate whether an authorized application submitted request since they’re included in every request
33. Need another mechanism that allows us to ensure that a request is authentic (coming from trusted page)
34. Four commonly used techniques
35. Referer validation
36. Secret Validation Token
37. Custom HTTP Header
38. sameSite Cookie