Recap so far

Recap: SOP vs JSON-P vs CORS

- SOP defines the concept of origin (protocol, hostname, port)
  - it restricts DOM access and **networks requests** to the origin
  - it does not restrict content inclusion (e.g., scripts)
  - it is more relaxed when dealing with cookies

- How to perform a network cross-origin request (A => B)?

  - **JSON-P** (hack): the idea is to include a script, whose content is dynamically generated by the B, that when executed by A will send data to one function from A. In practice, A can thus receive data from B.

  - **CORS** (proper way): the browser will allow A to read the response from B only if B explicitly whitelists A using a CORS header

Recap: cookies

Main attributes:

- **Domain**: when set, cookie can be accessed even by related domains. Hence, a cookie could be accessible by different origins from the same registrable domain.
- **Path**: when set, access to cookie is restricted based on the path
- **Secure**: when set, cookie is set only when using HTTPS
- **HttpOnly**: when set, javascript code cannot access cookie
- **SameSite**: whether a cookie is appended in case a cross-**site** request. Notice that site means a registrable domain (a.foo.com and b.foo.com have the same “site”: foo.com)

See slide #44 to understand how SOP is “adapted” in case of cookies (the browser will not reason on the SOP origin but will do way more).

Recap: problems of cookies

- Cookie Jar is organized based on (name, domain, path). However, its handling is browser specific:
  - **Cookie tossing attack**: subdomain A sets cookie X that can be accessed by subdomain B. What if B had already a cookie called X? The browser will define an “order” on the cookies and the attacker may exploit it.
  - **Cookie Jar Overflow**: given a cookie X with attribute HttpOnly, javascript code should not be able to change its value. However, an attacker may generate a large number of cookies, forcing the browser to discard the cookie X. Then, the attacker can arbitrarily set X using javascript code.
- When a cookie is sent in a request, only name=value is sent. Hence, the server is not aware and cannot verify the cookie attributes in the client (e.g., to reject the cookie if httpOnly is false).
- **Cross-site requests may leak the content of a cookie when SameSite is not set correctly.** We will some examples later on when we consider XSS.