Sessions and Cookies

Cookie Monster @MeCookieMonster

Me heard of something called “computer cookie”! How can me get one? Me bet it delicious!

280 RETWEETS    389 LIKES

7:42 AM - 13 Jan 2016
Review: Dynamic Pages

• Servers can **dynamically generate** content to send to a client
  • Backend software helps create HTML on the fly
  • **Server-side dynamic** pages require full reloads on the client
  • **Client-side dynamic** pages use JavaScript to modify the DOM without interacting with the server

• We can use **Flask** for Python to create various **routes** that allow dynamic creation of content
  • We use Flask by defining methods and **decorating** them with `@app.route()` calls to create **routes**
  • Flask’s structure **abstracts away** the filesystem – the **path** of a URL no longer refers to a **file** on the server, but encodes the **semantics of a request**
Review: JavaScript

- **JavaScript** is code embedded within HTML
  - JS lets you modify parts of the DOM tree
- JS is based around **asynchronous events**
  - Write a function that runs when:
    - the user clicks a mouse button
    - the user moves the mouse
    - the user presses a key
    - the page finishes loading
- You like JavaScript

```html
<html>
<head>
<body>
<button id="cool">Click me</button>
<script>
  document.getElementById("cool").onclick = function () {
    alert('hello'); }
</script>
</body></html>
```
Review: Python Decorators

• A **Decorator** is *syntactic sugar* in Python
• “@” prefix before something is a decorator
  • Decorators basically wrap functions in other functions

• Click library
  • Decorate a function with command line inputs that it can accept

• Flask library
  • Decorate functions that represent URL endpoints that get served
Review: Decorators

def add_message(fn):
    def inner():
        print("Calling...")
        fn()
    return inner

@add_message
def f():
    print("f ran")

f()  # "Calling..." followed by "f ran"
f()
One-Slide Summary: Sessions

• We use **sessions** to help maintain *state* between requests
  • Recall: HTTP is stateless, so we build sessions on top of the server and client to hack around this limitation

• **Sessions** are data stored on the *server*
  • Usually, some sort of map or dictionary structure; (key,value) pairs
  • e.g., uniqname=kjleach, cart=“1,2,3”

• **Cookies** are (key,value) pairs that are stored on the *client*
  • **Cookies** can help the server figure out which *session* is appropriate for a response
Sessions

• Sessions are used for
  • Knowing which user is logged in
  • Tracking the pages a user accesses
  • Remembering things in a shopping cart
  • **Web browser level**

• Not used for
  • Knowing which computer to send a response to
  • **Network level**
Why sessions are necessary

• Web requests use HTTP
• HTTP is "stateless": doesn't keep track of user who sent a request
  • (recall from Flask: there’s nothing built in that tracks an individual...)

• To allow for sessions (e.g. logging in), need a layer on top of HTTP
  • In Flask: your script will manage variables that help track individual users!
Sessions

• A session is a single "interaction" between the site and user
  • Precise definition depends on application

• Example: Facebook or Gmail login

• Example: Amazon cart
  • Even when you’re not logged in
Morton Salt, Iodized, 26 Ounce
by Morton

Price: $0.95 ($0.04 / Ounce) prime

- ALL PURPOSE - This all-purpose salt features uniformly shaped crystals making it the perfect SALT when precise measurements are critical
- GREAT FOR EVERYDAY USE - This an iodized salt, perfect for everything from cooking and baking to filling table salt shakers
- IODIZED - This salt supplies iodine, a necessary nutrient for proper thyroid functions
- VERSATILE - Meets various operational needs - both front and back of house
- EASY TO USE - Simplify your cooking needs in the kitchen with Morton all-purpose table salt

FREE Delivery by Sunday, July 12 for Prime members

Cart subtotal (9001 items): $0.95

Added to Cart

This is a gift
Thought Question

• If I open an "incognito mode" window in Chrome:
  • My Amazon shopping cart is empty in the incognito window
  • I can then log in as a second user in incognito
  • I'm still logged in as the original user in the non-incognito window

• How many sessions are active in this situation?

• Can you think of another situation where multiple sessions on the same computer is useful?
Session perspectives - client

Client

Put book in cart

OK

Put in another book

Server

(Modifies database)

(Modifies database again)
Session perspectives: server

• A server must track many sessions at once
• How can it tell the difference between clients?
Sessions and stateless HTTP

• HTTP is stateless, so we want to use a "session protocol" on top of it
  • JK there's no such thing as a “session protocol” #lol

• Implemented at application layer instead
  • State maintained in session variables
  • Data stored in one request can be accessed by later request
    • Caution: Flask is not durable – if you manage sessions with variables only, they’ll go away when you stop the script.
      • Use a DB instead

• Application layer sessions are one reason to use a web framework
  • Flask, Django, Ruby-on-rails, etc.
Implementing sessions

• Problems to solve:
  • Creating and destroying sessions
    • example: logging in/out
  • Linking sessions to users
    • I can be logged into Gmail on both my laptop and phone at same time
  • Linking HTTP requests to sessions
    • Keep track of individual browser tabs that might connect to the same website
When to create a session?

• Depends on the application

• Amazon?
  • When you visit the page
  • Needed for a shopping cart

• Gmail?
  • When you log in
  • Needed so that it shows the right mail to the right person

• Projects 2 and 3?
  • Wouldn’t it be nice to store whether someone logged in and not prompt on every request?
How to store session data?

• Session data storage is up to the server
• Best practice: store a small amount of data identify the session
  • Username, session ID, etc.
• Use session ID or username to do a database lookup
  • Shopping cart content, news feed items, etc.
When to close a session?

• We can't rely on logout
• Timeouts needed for almost all apps
  • When should the online game be reset?
  • When should Google forget your search?
  • When has your cart been abandoned?
  • When have you started searching for a different flight?
• Timeout from first request or most recent?
How to link sessions to users
How to link sessions to users

1. Client requests https://mail.google.com
   • Recall URL escaping: %3A == :, %2F == /
3. Client sends request with username and password
4. Server tests and responds with redirect to https://mail.google.com
Session implementation

• How to link HTTP requests to a session?

Turn this ... into this

<table>
<thead>
<tr>
<th>Client</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check credit card</td>
<td></td>
</tr>
<tr>
<td>Put book in cart</td>
<td></td>
</tr>
<tr>
<td>Put shirt in cart</td>
<td></td>
</tr>
<tr>
<td>Request gift wrapping</td>
<td></td>
</tr>
<tr>
<td>Take book out of cart</td>
<td></td>
</tr>
<tr>
<td>Put top hat in cart</td>
<td></td>
</tr>
<tr>
<td>Change address</td>
<td></td>
</tr>
<tr>
<td>Put in another book</td>
<td></td>
</tr>
</tbody>
</table>

Client                                Server

- Put book in cart                      (Modifies database)
  - OK
  - Put in another book                 (Modifies database again)
Cookies

• Cookies are small files on client machine
  • Carry state between HTTP requests
  • Contain key/value pairs

• According to the lore, the name originates from the story of Hansel and Gretel, who were able to mark their trail through a dark forest by dropping cookie crumbs behind them.
Cookies

Client  Server

- Put book in cart
- OK
- Put in another book
- (and set cookie)
- (Modifies database)
- (Modifies database again)
- (Cart looks full thanks to cookie info)
Example: cookies in Chrome

- See all cookies by browsing to `chrome://settings/siteData`
- Take a look at the cookies on your own laptop
Example: cookies in Chrome

• Browse to https://www.amazon.com/
• Settings / Advanced / Content settings / Cookies / See all cookies
Example: shopping cart

• Browse to https://www.amazon.com/
• Add something to the cart
Example: shopping cart

- Again, browse to https://www.amazon.com/
- Cart has one item, even though we’re on a different page
Example: shopping cart

- Clear cookies using the developer console
Example: shopping cart

• Again, browse to https://www.amazon.com/
• Cart appears empty
Cookie content

- **Name** is up to the server
- **Value** is up to server: encrypted? OK!
- **Domain** used by browser per-domain, total limits
- **Path** specifies scope of cookie
  - / vs. /cart/ or whatever
- **Expiration** tells client when to delete
- **Secure** is how cookie may be transmitted
Cookie content

• Cookies only over-writable by same domain and path
• Enforced by browser

• Store a username, session ID, etc. in the cookie
• Use session ID or username to do a database lookup
  • Shopping card content, news feed items, etc.
Cookie transfer: Server -> Client

• Headers include request to `Set-Cookie`
• These are exactly (some of) the cookies we saw in Chrome

```
$ curl --verbose --user-agent "Mozilla/5.0" https://www.amazon.com/ > /dev/null
...< Set-Cookie: skin=noskin; path=/; domain=.amazon.com
< Set-Cookie: session-id=130-5594428-2333702; Domain=.amazon.com;
Expires=Tue, 01-Jan-2036 08:00:01 GMT; Path=/
< Set-Cookie: session-id-time=20827872011; Domain=.amazon.com; Expires=Tue,
01-Jan-2036 08:00:01 GMT; Path=/
```
Cookie transfer: Client -> Server

$ curl --verbose --user-agent "Mozilla/5.0"
https://www.amazon.com/ ... > /dev/null
...
> HTTP 1.1 GET /
> Cookie: skin=noskin; session-id=130-5594428-2333702;
  session-id-time=2082787201l;
Cookie transfer

• Cookies add to HTTP overhead
• Again, best practice is to store a small amount of data in the cookie
• Use that data to do a database lookup on the server side
Saving cookies

• Your browser saves cookies by default

• **Tell curl to save cookies with --cookie-jar**

```bash
$ curl --verbose --user-agent "Mozilla/5.0"
    --cookie-jar cookies.txt https://www.amazon.com/
    > /dev/null

$ cat cookies.txt
    .amazon.com TRUE / FALSE 0 skin noskin
    .amazon.com TRUE / FALSE 2082787201 session-id 147-4402398-5757441
    .amazon.com TRUE / FALSE 2082787201 session-id-time 20827872011
```
Protecting cookies

• Anyone that has your cookies can convince server it's you

• **Session Hijacking** is when an adversary steals your session
  • If they can copy cookies from your computer, they can take your identify for those websites for which you have cookies downloaded
  • Consider: when you login, you get a cookie that the server uses to vet you as “already-logged in.”

• Use HTTPS so that Cookie headers aren’t sent in plaintext!
Encrypted cookie transfer

- If I have your cookies, I can steal your session
- To the server, I look just like you!
- Prevent this with cookies that can only be transmitted over HTTPS
Thought Question

• When you log into johnmail.com, it sets a cookie "userid=200".
• What might be insecure about this?
• How could you fix it?
Encrypted cookie values

• If the client has a copy of cookies set by the server, then the client can manipulate the server
• Prevent this with encrypted cookie content
  • Only the server can decrypt
  • **Note:** This is *separate* from using HTTPS to *transmit* the cookie!

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Domain</th>
<th>P...</th>
<th>Expires / Max...</th>
<th>Size</th>
<th>HTTP</th>
<th>Secure</th>
</tr>
</thead>
<tbody>
<tr>
<td>s-cbcsbTff</td>
<td>1</td>
<td>.amazon.com</td>
<td></td>
<td>2018-01-16T...</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a-main</td>
<td>Atza[wEBIL-ZoP_TnX5s5qVEdKvsoQR9M47Jqo0cT]_bIVnQG...</td>
<td>.amazon.com</td>
<td></td>
<td>2038-01-11T...</td>
<td>424</td>
<td></td>
<td></td>
</tr>
<tr>
<td>aws-ubold-main</td>
<td>102-1422888-1065586</td>
<td>.amazon.com</td>
<td></td>
<td>2096-02-03T...</td>
<td>32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>csm-hit</td>
<td>a-M2sBZTc3VeF120DMyJ4DF6j1516118415619</td>
<td><a href="http://www.amazon.com">www.amazon.com</a></td>
<td></td>
<td>2018-01-23T...</td>
<td>43</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ic-main</td>
<td>en_US</td>
<td>.amazon.com</td>
<td></td>
<td>2038-01-11T...</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sess-ai-main</td>
<td>&quot;biYLsevGv15ySi5RDUGioDQGcU9+k+PLiNOMI3JoPcPj2=&quot;</td>
<td>.amazon.com</td>
<td>Session</td>
<td>58</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>session-id</td>
<td>1-42-6976303-8298237</td>
<td>.amazon.com</td>
<td></td>
<td>2036-01-01T...</td>
<td>29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>session-id-time</td>
<td>20827872011</td>
<td>.amazon.com</td>
<td></td>
<td>2036-01-01T...</td>
<td>26</td>
<td></td>
<td></td>
</tr>
<tr>
<td>session-token</td>
<td>OCFp6c+N1p2eULwZI+3wUqJh5F3i6Jn5mC19oN5kve...</td>
<td>.amazon.com</td>
<td></td>
<td>2038-01-11T...</td>
<td>281</td>
<td></td>
<td></td>
</tr>
<tr>
<td>skin</td>
<td>nookin</td>
<td>.amazon.com</td>
<td></td>
<td>Session</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>spblockdate</td>
<td>1514191783143</td>
<td>.amazon.com</td>
<td></td>
<td>2027-12-31T...</td>
<td>24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ssi-main</td>
<td>Ss111PQg1LW9hWUJZrPbl+PiP8CAsBYdLm3z4PvGgEbo...</td>
<td><a href="http://www.amazon.com">www.amazon.com</a></td>
<td></td>
<td>2038-01-11T...</td>
<td>295</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ubid-main</td>
<td>123-7833599-2974057</td>
<td>.amazon.com</td>
<td></td>
<td>2036-01-01T...</td>
<td>28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x-main</td>
<td>&quot;J1bFtw6Y%0bUWQ9AtJbP3PI8Sb7kZ0FMYb9bZ0OC77y...</td>
<td>.amazon.com</td>
<td></td>
<td>2038-01-11T...</td>
<td>72</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Session vs. permanent cookies

- Session cookies are deleted by the client shuts down
  - Close the tab or quit the browser
- Permanent cookies have explicit expiration dates

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Domain</th>
<th>Path</th>
<th>Expires/Max-Age</th>
<th>Size</th>
<th>HTTP</th>
<th>Secure</th>
</tr>
</thead>
<tbody>
<tr>
<td>s-qpcbrf</td>
<td>1</td>
<td>.amazon.com</td>
<td>/</td>
<td>2018-01-16T...</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sl-main</td>
<td>AtzaiwEBIL-ZdP_TnX55gV8EdKvsoGRMnTJoq0o7uWnQG...</td>
<td>.amazon.com</td>
<td>/</td>
<td>2038-01-11T...</td>
<td>424</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>aws-ubid-main</td>
<td>102-1422888-1065586</td>
<td>.amazon.com</td>
<td>/</td>
<td>2086-02-03T...</td>
<td>32</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>csm-hit</td>
<td>s-M2BVZTSVh5F120DMVYJDF615156415619</td>
<td><a href="http://www.amazon.com">www.amazon.com</a></td>
<td>/</td>
<td>2018-01-23T...</td>
<td>43</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ic-main</td>
<td>en_US</td>
<td></td>
<td></td>
<td>2038-01-11T...</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sess-ut-main</td>
<td>&quot;bWyhexGV15YS1sRDUGi8DGd6hk+PLNN6M3JPNaPj28X&quot;</td>
<td>.amazon.com</td>
<td>/</td>
<td>Session</td>
<td>58</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>session-id</td>
<td>142-6576303-8280237</td>
<td>.amazon.com</td>
<td>/</td>
<td>2036-01-01T...</td>
<td>29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>session-id-time</td>
<td>20827827011</td>
<td></td>
<td></td>
<td>2036-01-01T...</td>
<td>26</td>
<td></td>
<td></td>
</tr>
<tr>
<td>session-token</td>
<td>OCqpc+N1p2eJwVrZi3wUqsR6Fm36JhYjSMrC19oN5kve...</td>
<td>.amazon.com</td>
<td>/</td>
<td>2038-01-11T...</td>
<td>261</td>
<td></td>
<td></td>
</tr>
<tr>
<td>skin</td>
<td>noskin</td>
<td>.amazon.com</td>
<td></td>
<td>Session</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>spblockdate</td>
<td>1514918331X3</td>
<td><a href="http://www.amazon.com">www.amazon.com</a></td>
<td>/</td>
<td>2027-12-31T...</td>
<td>24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ssl-main</td>
<td>Sssl1[+QGJ[LUW9MVJZP1BiP1XCH8CAeAYuLmz4cPv9GbeQ...</td>
<td>.amazon.com</td>
<td>/</td>
<td>2038-01-11T...</td>
<td>295</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ubid-main</td>
<td>152-7533599-2974057</td>
<td>.amazon.com</td>
<td>/</td>
<td>2038-01-01T...</td>
<td>26</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x-main</td>
<td>&quot;JqL@8cu6yH10xwuWqR4flePY9l8991zGfYX08bDQgC7fV...</td>
<td>.amazon.com</td>
<td>/</td>
<td>2038-01-11T...</td>
<td>72</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x-wu-ulid</td>
<td>1qpmQzrHyzowtwh6DHEBwU08fYf4FGX+05T9dYLEygZC8kyl...</td>
<td>.amazon.com</td>
<td>/</td>
<td>2036-01-01T...</td>
<td>149</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Third-party cookies

- Page may contain objects from many sources
  - Scripts, images, etc.
- These 3rd-party objects set and get cookies
- Example: nytimes.com

```html
<html>
<head>
  <script
    src="https://tags.bluekai.com/site/50550?ret-js&limit=1"
    type="text/javascript"
  >
  </script>
</head>
<body>
  <img src="https://static01.nyt.com/newsgraphics/2018/01/18/shutdown-deal-factions/assets/images/lindsey-graham.png">
</body>
</html>
```
Third-party cookies

- Cookies have a domain
- **First-party cookie**: domain is the same as the domain of the page you are on
- **Third-party cookie**: domain is different
- Example from nytimes.com

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Domain</th>
<th>Expires / MaxAge</th>
<th>Size</th>
<th>HTTP</th>
<th>Secure</th>
<th>SameSite</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDE</td>
<td>AHWqTUJe_KCi0XNhNt3LWAFpbPHkNNCKdka8oVRxFD</td>
<td>.doubleclick.net</td>
<td>2020-01-16</td>
<td>67</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UID</td>
<td>15718486243203196565e8cg1516118885</td>
<td>.scorecardres...</td>
<td>2020-01-06</td>
<td>36</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UIDR</td>
<td>1516118885</td>
<td>.scorecardres...</td>
<td>2020-01-06</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_gads</td>
<td>ID=a238e066eb072b:T=1516118885/S=ALNI_My9...</td>
<td>nytimes.com</td>
<td>2020-01-16</td>
<td>75</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>__utmz</td>
<td>69104142.228179004.1516118896.1516118895.1516118895.15161...</td>
<td>nytimes.com</td>
<td>2018-01-16</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>__utmc</td>
<td>69104142</td>
<td>nytimes.com</td>
<td>Session</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>__utmt</td>
<td>1</td>
<td>nytimes.com</td>
<td>2018-01-16</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>__utmz</td>
<td>69104142.1516118896.1.1.utmcarr=</td>
<td>d</td>
<td>utmccon=(cli...</td>
<td>nytimes.com</td>
<td>2018-07-18</td>
<td>75</td>
<td></td>
</tr>
</tbody>
</table>
Example: Google third party cookies

- One of the main advertising cookies on non-Google sites is named **IDE** and is stored in browsers under the domain **doubleclick.net**.
  - [https://www.google.com/policies/technologies/types/](https://www.google.com/policies/technologies/types/)
- Visit nytimes.com and view cookies:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Domain</th>
<th>P...</th>
<th>Expires / Max...</th>
<th>Size</th>
<th>HTTP</th>
<th>Secure</th>
<th>SameSite</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDE</td>
<td>AHYq7JUe_KC101niN13LWAFbPHIIkNfKdka5oVRxF...</td>
<td>.doubleclick.net</td>
<td>/</td>
<td>2020-01-16...</td>
<td>67</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UID</td>
<td>15718436a24320a160668q1516118895</td>
<td>.scorecardres...</td>
<td>/</td>
<td>2020-01-06...</td>
<td>36</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UIDR</td>
<td>1516118895</td>
<td>.scorecardres...</td>
<td>/</td>
<td>2020-01-06...</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>__gads</td>
<td>ID=2438c63e066e0b072bT=1516118895:S=ALNI_MYY9...</td>
<td>.nytimes.com</td>
<td>/</td>
<td>2020-01-16...</td>
<td>75</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>__utma</td>
<td>69104142.228179094.1516118895.1516118895.15161...</td>
<td>.nytimes.com</td>
<td>/</td>
<td>2020-01-16...</td>
<td>59</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>__utmz</td>
<td>69104142.1.10.1516118896</td>
<td>.nytimes.com</td>
<td>/</td>
<td>2018-01-16...</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>__utmz</td>
<td>69104142</td>
<td>.nytimes.com</td>
<td>/</td>
<td>Session</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>__utmz</td>
<td>1</td>
<td>.nytimes.com</td>
<td>/</td>
<td>2018-01-16...</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>__utmz</td>
<td>69104142.1516118895.1.1.utmcsr=(direct),utmomcr=(dir...</td>
<td>.nytimes.com</td>
<td>/</td>
<td>2018-07-16...</td>
<td>75</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example: Google third party cookies

• You type nytimes.com into your browser
• Browser issues **GET** request to nytimes.com
  • Includes nytimes.com cookies
• Browser receives HTML for nytimes.com
  • HTML includes some JavaScript via the `<script>` tag
• Browser executes JavaScript included by nytimes.com
  • JS code figures out you are on nytimes.com, e.g., with `window.location.href`
  • JS codes initiates a request to doubleclick.net
• Browser issues **GET** request to doubleclick.net
  • Includes doubleclick.net cookie
  • Appends your current location (nytimes.com) to the URL
• Now, doubleclick.net (AKA Google) knows you visited nytimes.com
Who uses third-party cookies?

• Companies that sell ads directly
  • Google and Facebook

• Companies that sell information about you
  • Acxiom "one of the biggest companies you've never heard of" ($1B+)

• “If you're not paying for the product, then you are the product”
  • That's mostly how the business side of the web is structured
Checking what you send to trackers

• [https://panopticlick.eff.org/](https://panopticlick.eff.org/)
Browser fingerprinting

• Browser fingerprinting attempts to uniquely identify your browser using information other than cookies
  • User Agent
  • Time zone
  • Fonts
  • Language
  • And lots more

• Anti-fingerprinting options announced by Firefox, Chrome, Safari
  • As of fall 2019
Avoiding trackers

• Add-ons like uBlock Origin, Privacy Badger, Brave, Disconnect or ScriptNo block trackers
• Monitors embedded links and blacklists trackers
Discussion

• Many web companies make their money from information about users
• In exchange, they give you a "free" service (email, web search, a platform for gossip, etc.)
• Is it OK to block trackers?
import flask
c
app = flask.Flask(__name__)

app.secret_key = b'uAy\x9d\x08{\x12\x8d\x9d\x1f\xbar\x86A\x9fpQy4\x05)v04'

def index():
    if "user" in flask.session:
        user = flask.session["user"]
        app.logger.debug("Get user=\%s", user)
        return "<html><body>Hello {}</body></html">.format(user)
    else:
        flask.session["user"] = "awdeorio"
        app.logger.debug("Set user=\%s", user)
        return "<html><body>Logging in ...</body></html>"

if __name__ == '__main__':
    app.run(debug=True)
Cookie example

- Start server
  
  ```
  $ python3 test.py
  * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
  127.0.0.1 -- [22/Jan/2019 08:40:31] "GET / HTTP/1.1" 200 -
  ```

- Browse to http://localhost:5000/

  ![Login screen](image)

- See encrypted cookie in developer console

  ![Cookie table](image)
import flask
app = flask.Flask(__name__)

app.secret_key = b'uAy\x9d\x08[\x12\x8d\x9d\x1f\xbar\x86A\x9fpQy4\x05)v04'

# ...

• Session cookies are encrypted
• The server has the encryption key
Cookie encryption key

• How to generate a secure encryption key?

• Need a cryptographically secure random number.
  • If a "random" number is in any way predictable, then a hacker could guess it and masquerade as any user!

• Generate a Python string

• $ python3 -c "import os; print(os.urandom(24))"
  b'uAy\x9d\x08[\x12\x8d\x9d\x1f\xbar\x86A\x9fpQy4\x05)v04'
Further reading

• Good reference on cookies
• https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies