Session Management

Lecture 10

Topics

We will look at

- Cookies - client-side storage
- Details of Session implementation
- Using Sessions in Node.js (using connect)

Revise the Guessing Game application to keep the correct answer on the server

We'll also discuss HTML5's local storage

Cookies

Often a web site wants know if you've visited their page before, and perhaps what you did last time

Maybe a shopping cart ID that ties to their database

While the server can obtain your IP address…

- There is no guarantee that the same IP address means the same user
- … or that the same user always accesses the site with the same computer
- And importantly, most machines' IP addresses actually change over time…

Cookies

To partially solve this problem, a standard emerged that lets the server **push** name/value pairs on to the client's (browser) machine.

The browser can ignore them. But if its cooperating, it will resend the name/value pairs with each request.

These name/value pairs are called **cookies**.

Cookie implementation

A server sends cookies back to the browser through Response Headers

```
HTTP/1.0 200 OK
Content-type: text/html
Set-Cookie: name=value
Set-Cookie: name2=value2; Expires=Wed, 09 Jun 2021 10:18:14 GMT

(content of page)
```

- A browser can store these cookies in the file system, or using any other means necessary.
- Importantly, they are associated with the domain name of the sending server (www.example.com)

Cookie implementation

Each time the browser sends a request to example.com, it will look up any cookies from that site.

It will send them in the request header

```
GET /spec.html HTTP/1.1
Host: www.example.org
Cookie: name=value; name2=value2
Accept: */*
```

Cookies - good and bad

There is nothing inherently “bad” about cookies - in fact its quite difficult to fully use the web without them…

- When used “honestly” they can improve UX
- The browser (user) has control - cookies can be deleted or simply not sent.

Cookies - good and bad

However - there are privacy issues.

1. If someone can see cookies your browser collected, they can see your internet history.
    a. This can be mitigated by security permissions, encryption, etc.
2. Cookies can let ad servers track you if you aren't careful.
3. Cookies can be hijacked, and if the server trusts them unconditionally, can allow you to be hacked

Ads...

Recall that an image, embedded in an HTML page, causes a secondary HTTP request to execute

referer is an important HTTP request header

Whenever a request is initiated by clicking on a link, or by an <img>, <video>, or <audio> tag, the URL of the original page is sent with the new request

This is generally nice, and lets people collect money for referrals, etc.

Ads

Many (legitimate) web sites use ads (usually img, video, or iframe elements).

- These ads are served from other servers, operated by advertising companies.
- Many web sites **share** ad services, and thus ad servers.
- If site A and B use Ad Server C, then each time you visit site A and B you make requests to C.
- From the referer header, C can determine where the request came from.

Ads

So here is the problem:

- When the ad server ad.com gets a request, it sets a cookie on its response (id=67).
- ID 67 points to a database record that stores each **referer** value that came in for user 67.
- Each time you visit a site that has embedded ads from ad.com, you send the ad.com cookie with the request, and a new referer value.
- ad.com now has a nice record of the pages you visit, especially if its ads are everywhere on the net!

You would be surprised just how few major ad players there are… which means they ARE everywhere...

Hacking

- You may or may not want ad server to know your history… but most browsers will let you decide
  - 3rd party cookies can be blocked in most browsers

- However - what if a cookie stored an authentication token to let you bypass entering your username/password?
  - Now anyone with that cookie could send it to the server and bypass security!

Hacking

- How could someone get your cookie data?
  - Physical access to the disk/machine
  - Network Eavesdropping
    - HTTP is plain text!

This really is the server's responsibility. You (as a server dev) should never rely on cookies too much

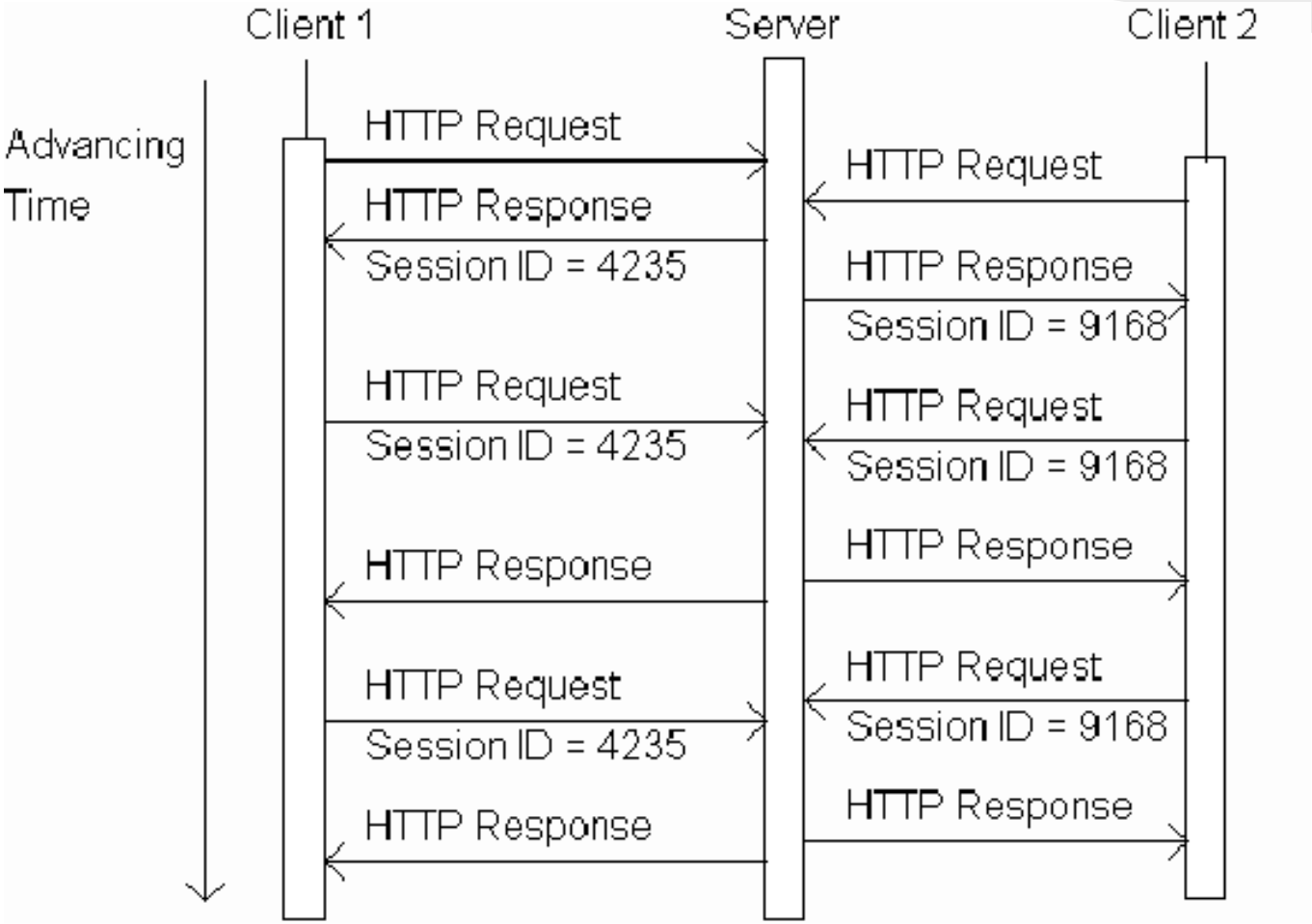
- Re-authenticate before displaying very sensitive data (credit card information)
- Use SSL to encrypt the HTTP traffic.

Sessions

Cookies are stored client-side, but they typically *point* to some resource on the server (a database record for ID=67)

- Sessions are data objects that can be associated with a user.
- The objects exist **only** on the server (memory, database, etc)
- However, the ID for the session is sent as a cookie, letting the server "lookup" the session data for each request.

Session Data



Session Data

- Sessions are optional, because cookies are optional!
  - A server can't rely on the session id being in the request!
- Sessions typically *expire*
  - When the server gets a request with an expired session id, it creates a new session object

- You don't need to worry too much about the data you store **in** the session - its on the server, so its secure, and doesn't cost you in bandwidth....
  - Of course, if you display the session data on a web page - all bets are off!

Enabling Sessions

- We could implement sessions ourselves (by using the cookie headers)- but that's silly…
- Happily, the connect middleware package contains session management functionality
  - http://www.senchalabs.org/connect/session.html

```
connect()
  .use(connect.cookieParser())
  .use(connect.session(
      { secret: 'secret'}))
```

Setting session data

```
if ( !req.session.number) {
    req.session.number = 1;
    console.log("New Session");
}
else {
    req.session.number++;
    console.log("Session number = " + req.session.number);
}
```

Guessing Game

On a request for start.html, we'll create a random number and **store it in the session**.

We'll remove the hidden form implementation part.

We can now record a list of all guesses, and the results for each guess.

HTML5 Local Storage

- As a side note - cookies are no longer the only way to store data on the client machine.
- HTML5 introduces client-side, local storage.
- It is an object (set of name value pairs) directly accessible with **client-side** JavaScript
- We will see it in action later - but for now its important that you understand the critical differences between cookies and local storage

Next Lecture

We move to formatting our HTML to make it look more professional.

Cascading Style Sheets

- Read HTML text book, chapters 7-9